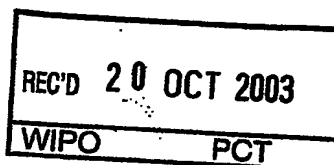




PCT/FR 03 / 0 2245



BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 12 MAI 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS
CONFORMÉMENT À LA
RÈGLE 17.1.a) OU b)

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersburg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr



26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

BREVET D'INVENTION CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI



N° 11354*03

REQUÊTE EN DÉLIVRANCE

page 1/2



Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 0 1 / 210502

REMISE DES PIÈCES DATE 28 NOV 2002 LIEU 75 INPI PARIS N° D'ENREGISTREMENT 0214964 NATIONAL ATTRIBUÉ PAR L'INPI DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 28 NOV. 2002		NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE CABINET NETTER 36 avenue Hoche 75008 PARIS	
Vos références pour ce dossier (facultatif) INRIA Aff. 59-PI (120813)			
Confirmation d'un dépôt par télécopie		<input type="checkbox"/> N° attribué par l'INPI à la télécopie	
2 NATURE DE LA DEMANDE		Cochez l'une des 4 cases suivantes	
Demande de brevet		<input checked="" type="checkbox"/>	
Demande de certificat d'utilité		<input type="checkbox"/>	
Demande divisionnaire		<input type="checkbox"/>	
Demande de brevet initiale		N°	Date
ou demande de certificat d'utilité initiale		N°	Date
Transformation d'une demande de brevet européen		<input type="checkbox"/>	Date
Demande de brevet initiale		N°	Date
1 TITRE DE L'INVENTION (200 caractères ou espaces maximum) Compression de données numériques robuste au bruit de transmission.			
4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE		Pays ou organisation France Date 2 2 0 7 2 0 0 2 N° 02 09287 Pays ou organisation Date N° Pays ou organisation Date N° <input type="checkbox"/> S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»	
5 DEMANDEUR (Cochez l'une des 2 cases)		<input type="checkbox"/> Personne morale <input type="checkbox"/> Personne physique	
Nom ou dénomination sociale		INRIA INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE	
Prénoms			
Forme juridique		Etablissement Public National à caractère scientifique et technologique	
N° SIREN			
Code APE-NAF			
Domicile ou siège	Rue	Domaine de Voluceau - BP 105	
	Code postal et ville	7 8 1 5 3 Rocquencourt	
	Pays	France	
Nationalité		française	
N° de téléphone (facultatif)		N° de télécopie (facultatif)	
Adresse électronique (facultatif)			
<input checked="" type="checkbox"/> S'il y a plus d'un demandeur, cochez la case et utilisez l'imprimé «Suite»			

Remplir impérativement la 2^{ème} page

Reservé à l'INPI

REMISE DES PIÈCES
DATE **28 NOV 2002**
LIEU **75 INPI PARIS**
N° D'ENREGISTREMENT **0214964**
NATIONAL ATTRIBUÉ PAR L'INPI

DB 540 W / 210502

6 MANDATAIRE (s'il y a lieu)		
Nom	PLAÇAIS	
Prénom	Jean-Yves	
Cabinet ou Société	Cabinet NETTER	
N° de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	36 avenue Hoche
	Code postal et ville	75 008 PARIS
	Pays	France
N° de téléphone (facultatif)	01 58 36 44 22	
N° de télécopie (facultatif)	01 42 25 00 45	
Adresse électronique (facultatif)		
7 INVENTEUR (S)		Les inventeurs sont nécessairement des personnes physiques
Les demandeurs et les inventeurs sont les mêmes personnes		<input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non : Dans ce cas remplir le formulaire de Désignation d'Inventeur(s)
8 RAPPORT DE RECHERCHE		Uniquement pour une demande de brevet (y compris division et transformation)
Établissement immédiat ou établissement différé		<input checked="" type="checkbox"/> <input type="checkbox"/>
Paiement échelonné de la redevance (en deux versements)		Uniquement pour les personnes physiques effectuant elles-mêmes leur propre dépôt <input type="checkbox"/> Oui <input type="checkbox"/> Non
9 RÉDUCTION DU TAUX DES REDEVANCES		Uniquement pour les personnes physiques <input type="checkbox"/> Requête pour la première fois pour cette invention (joindre un avis de non-imposition) <input type="checkbox"/> Obtenue antérieurement à ce dépôt pour cette invention (joindre une copie de la décision d'admission à l'assistance gratuite ou indiquer sa référence) : AG <input type="checkbox"/>
10 SÉQUENCES DE NUCLEOTIDES ET/OU D'ACIDES AMINÉS		<input type="checkbox"/> Cochez la case si la description contient une liste de séquences
Le support électronique de données est joint		<input type="checkbox"/>
La déclaration de conformité de la liste de séquences sur support papier avec le support électronique de données est jointe		<input type="checkbox"/>
Si vous avez utilisé l'imprimé «Suites», indiquez le nombre de pages jointes		3
11 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) N° Conseil 92-1197 (B) (M) Jean-Yves PLAÇAIS		VISA DE LA PRÉFECTURE OU DE L'INPI M. ROCHET

Compression de données numériques robuste au bruit de transmission

5 L'invention concerne la compression de données numériques, notamment pour des signaux multimédia (audio, image, vidéo, parole), et la transmission robuste de ces données sur des réseaux bruités, tels que des réseaux sans fils, des communications mobiles.

10 Pour réduire le débit de transmission des données numériques, on les comprime, en cherchant à s'approcher du maximum théorique que les spécialistes appellent "l'entropie du signal". Pour cela, on utilise souvent des codes statistiques aussi appelés codes à longueurs variables, par exemple les codes de Huffman. Ces codes ont néanmoins l'inconvénient d'être très sensible aux erreurs de transmission. L'inversion de un bit peut conduire à une désynchronisation du décodeur, qui a pour conséquence un décodage erroné de toutes les données qui suivent la position du bit erroné.

15

Les solutions actuelles de compression, transmission et décompression de signaux multimédia sur réseau, sur lesquelles on reviendra, se fondent sur l'hypothèse qu'une certaine qualité de service du transport des données est garantie. En d'autres termes, elles supposent que les couches de transport et de liaison, en s'appuyant sur l'utilisation de codes
20 correcteurs, vont permettre d'atteindre un taux d'erreur résiduel quasi nul (c'est-à-dire vu par l'application de compression et de décompression). Mais cette hypothèse de taux d'erreur résiduel quasi nul n'est plus vraie lorsque les caractéristiques des canaux varient dans le temps (canaux non stationnaires), notamment dans les réseaux sans fil et mobiles, et pour une complexité du code de canal réaliste. En outre, l'ajout de redondance par codes
25 correcteurs d'erreurs conduit à réduire le débit utile.

Le besoin se fait donc sentir de solutions qui soient robustes au bruit de transmission, c'est à dire qui soient peu affectées par les erreurs de bits induites par ce bruit d'une part et qui permettent une utilisation optimale de la bande passante (c'est à dire de la capacité) du
30 réseau d'autre part.

La présente invention vient proposer des avancées en ce sens.

Selon un aspect de l'invention, il est proposé un codeur de compression de données numériques, qui comprend:

- une entrée (physique ou non) pour un premier flux de données, et un second flux de données,
- 5 - un module de codage, établissant une correspondance entre des symboles du premier flux de données et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole, et
- un module de traitement pour coder les symboles du premier flux de données d'après la correspondance, en choisissant parmi les mots redondants, en fonction d'une partie au moins
- 10 du second flux de données.

Selon différents autres aspects:

- les mots de code peuvent être de longueur fixe,
- le module de traitement comprend:
 - 15 . une fonction de calcul de la capacité de multiplexage courante du premier flux de données, au vu de la table de codage, et
 - . une fonction d'extraction, dans le second flux de données, d'une partie multiplexée, déterminée d'après la capacité de multiplexage courante, pour être portée par lesdits mots redondants.
- 20 - le codeur comprend une transformation d'un flux binaire en un flux de variable multi-valuée, en particulier en utilisant les transformations décrites dans la table C ci-après.
- en variante, le codeur comprend une transformation d'un flux binaire en un flux de variable multi-valuée, en particulier en utilisant une décomposition euclidienne généralisée à partir d'une variable globale donnée par la relation (E9) décrite ci-après.

25

Selon une première variante :

- le module de codage comprend une table de codage et le module de traitement comprend:
 - une fonction de lecture d'une capacité de multiplexage de chaque symbole courant du premier flux de données au vu de la table de codage, et
 - 30 - une fonction d'extraction d'une partie du second flux de données déterminée d'après la capacité de multiplexage, pour être portée par lesdits mots redondants,
- la table de codage comprend pour chaque symbole, un nombre de mots de code associé égal à une puissance de 2.

Selon une deuxième variante :

- le module de codage comprend un arbre binaire de codage contenant, pour chaque symbole du premier flux de données, une première partie de mot de code, de longueur variable et inférieure à une longueur maximale, et le module de traitement comprend:

- 5 - une fonction de calcul de la capacité de multiplexage pour chaque symbole courant du premier flux de données au vu de la première partie de mot de code de chaque symbole,
- une fonction d'extraction d'une partie du second flux de données déterminée d'après la capacité de multiplexage, pour être portée par lesdits mots redondants.
- 10 - en variante, chaque symbole comprend une séquence de symboles.

Selon une troisième variante :

- chaque symbole comprend une séquence de symboles, et le module de codage comprend un codeur arithmétique propre à calculer, pour une séquence de symboles du premier flux de données, une première partie de mot de code, de longueur variable et inférieure à une longueur maximale; le module de traitement comprend:

- une fonction de calcul de la capacité de multiplexage pour chaque symbole courant du premier flux de données au vu de la première partie de mot de code de chaque symbole,
- 20 - une fonction d'extraction d'une partie du second flux de données déterminée d'après la capacité de multiplexage pour chaque symbole, pour être portée par lesdits mots redondants.

25 Dans la deuxième et troisième variante, ladite partie du second flux de données est concaténée à la première partie de mot de code jusqu'à la longueur maximale du mot de code.

De manière générale:

- le second flux de données est préalablement encodé.
- 30 - le reste du second flux de données est concaténé aux données transmises.

L'invention vise également un décodeur, propre à effectuer les opérations inverses ou réciproques de celle du codeur, dans ses différents aspects.

L'invention vise encore un procédé de compression de données numériques, qui comprend les étapes suivantes:

- a. établir une correspondance entre des symboles du premier flux de données et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole, et
 - b. coder les symboles d'un premier flux de données d'après l'étape a., en choisissant parmi les mots redondants, en fonction d'une partie au moins d'un second flux de données.
- 10 Ce procédé peut intégrer les autres aspects du codage.

Enfin, l'invention vise également le procédé de décompression de données numériques, comprenant les étapes réciproques de celles du procédé de compression.

- 15 D'autres caractéristiques et avantages de l'invention apparaîtront à l'examen de la description détaillée ci-après, et des dessins annexés, sur lesquels :
- la figure 1 illustre de façon schématique un procédé de création de codes,
 - la figure 2 est un diagramme formant vue générale du processus d'encodage dans sa variante principale,
 - la figure 2A illustre un exemple simplifié de code multiplexé, à quatre éléments,
 - la figure 3 illustre un premier mode de réalisation d'un processus détaillé d'encodage,
 - la figure 4 illustre une variante de réalisation du processus de la figure 3,
 - la figure 5 illustre un exemple de création d'une capacité de stockage par l'attribution de plusieurs mots de code à un symbole, un flux de données q pouvant être conjointement stocké, et
 - la figure 6 ou table C illustre des transformations utilisées dans un exemple où le paramètre f_v est égal à 5,
 - la figure 7 illustre un autre exemple simplifié de code multiplexé à quatre éléments,
 - la figure 8 illustre un processus d'encodage général des figures 3 et 4,
 - la figure 9 illustre un deuxième mode de réalisation d'un processus détaillé d'encodage,
 - la figure 10 illustre une table de résultat d'une première variante du deuxième mode de réalisation du processus d'encodage selon la figure 9,

-la figure 11 illustre un diagramme formant vue générale d'une deuxième variante du deuxième mode de réalisation du processus d'encodage selon la figure 9

- la figure 12 un diagramme formant vue générale d'une troisième variante du deuxième mode de réalisation du processus d'encodage selon la figure 9,

5 En outre:

- l'annexe 1 contient des expressions utilisées dans la présente description, et

- l'annexe 2 contient des algorithmes en langage naturel utilisés dans la présente description.

10 Les dessins et les annexes à la description comprennent, pour l'essentiel, des éléments de caractère certain. Ils pourront donc non seulement servir à mieux faire comprendre la description, mais aussi contribuer à la définition de l'invention, le cas échéant.

De manière générale, un système de compression de signaux multimédia (image, vidéo, audio, parole) fait appel à des codes statistiques aussi appelés codes à longueurs variables.

15 Ceux-ci permettent d'obtenir des débits approchant ce que les spécialistes appellent "l'entropie du signal". Les codes les plus utilisés dans les systèmes existants (en particulier dans les standards) sont les codes de Huffman qui ont été décrits dans l'ouvrage suivant : D.A. Huffman : "A method for the construction of minimum redundancy codes", Proc. IRE, 40 (1951), p.1098-1101.

20

Plus récemment, on a vu un regain d'intérêt pour les codes arithmétiques en raison de leurs performances accrues en terme de compression. Ils permettent en effet de découpler le processus d'encodage du modèle supposé de la source. Ceci permet aisément d'utiliser des modèles statistiques d'ordre supérieur. Ces codes arithmétiques ont été décrits dans des

25 ouvrages tels que

- J.J. Rissanen "Generalized kraft inequality and arithmetic", IBM J.Res. Develop., 20:198-203, Mai 1976

- J.J. Rissanen, "Arithmetic Coding as number representations", Acta Polytech. Scand. Math., 31:44-51, Décembre 1979

30 ainsi que dans des brevets américains US 4 286 256, US 4 467 317, US 4 652 856.

Jusque récemment, la conception des systèmes de compression se faisait en supposant une qualité de service transport garantie. On supposait en effet que les couches inférieures du

modèle OSI incorporent des codes correcteurs d'erreurs garantissant un taux d'erreur résiduel vu de l'application quasi-nul.

5 Les codes à longueurs variables pouvaient donc être largement utilisés malgré leur forte sensibilité au bruit de transmission. Toute erreur dans le train binaire peut engendrer une désynchronisation du décodeur et donc une propagation des erreurs sur la suite des informations décodées.

10 Pour pallier ce problème de propagation, les standards de premières générations (H.261, H.263, MPEG-1, MPEG-2) ont incorporé dans la syntaxe du train binaire transmis des marqueurs de synchronisation. Ce sont des mots de code longs (16 ou 22 bits constitués d'une suite de 15 ou 21 bits à '1' suivis d'un '0') non émulables par des erreurs se produisant sur les autres mots de code et qui peuvent donc être reconnus par le décodeur avec une probabilité proche de '1'.

15 Cela conduit à structurer le train binaire en paquets délimités par ces marqueurs de synchronisation. Cela permet de confiner la propagation des erreurs au sein du paquet. Cependant si une erreur intervient en début de paquet la suite du paquet peut être perdue. En outre, la périodicité de ces marqueurs de synchronisation doit être restreinte pour éviter une
20 perte trop grande en efficacité de compression.

Cette hypothèse de taux d'erreur résiduel quasi nul n'est plus vraie dans les réseaux sans fil et mobiles, dont les caractéristiques des canaux varient dans le temps (canaux non stationnaires). Ce taux d'erreur résiduel vu par le décodeur des signaux de source est souvent
25 loin d'être négligeable.

Les nouveaux standards (H.263+ et MPEG-4) ont alors fait appel à des codes à longueurs variables réversibles (RVLC). La particularité de ces codes est qu'ils peuvent être décodés du premier vers le dernier bit d'un paquet, et, à l'inverse, du dernier vers le premier bit du
30 paquet.

Si une erreur s'est produite en milieu de paquet, cette symétrie du code permet de confiner la propagation des erreurs sur un segment en milieu de paquet au lieu de la propager jusqu'à

la fin du paquet délimité par un marqueur de synchronisation. Cependant, la symétrie du code engendre une perte en efficacité de compression par rapport à un code de Huffman de l'ordre de 10 %. En outre les codes à longueurs variables réversibles n'évitent pas complètement le problème de propagation des erreurs : si une erreur se produit en début et

5 en fin de paquet, tout le paquet risque d'être erroné.

La conception de codes qui soient à la fois performants en compression (i.e., qui permettent d'approcher l'entropie de la source), tout en étant robustes au bruit de transmission, constitue donc un enjeu important, notamment pour les futurs systèmes de communication multimédia

10 (image, vidéo, audio, parole) mobiles. Pour ces systèmes de nouveaux standards sont à l'étude à la fois au sein de l'ITU (International Telecommunication Union) et de l'ISO (International Standard Organization).

Bien que les standards occupent une place prépondérante dans le secteur des télécommunications, une telle famille de codes peut aussi trouver des applications sur des marchés de

15 niche faisant appel à des solutions propriétaires.

De façon générale, les procédés de compression de données numériques proposés ci-après sont mis en oeuvre par un codeur de compression de données numériques selon l'invention

20 comprenant un module de codage et un module de traitement.

Plus spécifiquement, le module de codage établit une correspondance entre des symboles d'un premier flux de données et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole. De manière générale, un

25 module de codage peut être toute forme de stockage présentant la correspondance définie ci-dessus, toute forme de représentation de cette correspondance ou toute fonction de codage calculant cette correspondance. Ainsi, à titre d'exemple seulement, le module de codage peut être une table de codage, un arbre de codage ou un codeur arithmétique suivant les procédés de compressions détaillés ci-après.

30

Le module de traitement met en oeuvre les étapes des procédés de compression de données à partir du premier flux de données et d'un second flux de données et du module de codage. Le module de traitement comprend des fonctions propre à réaliser certaines étapes des

procédés et peut comprendre une transformation d'un flux binaire en un flux de variable multi-valuée. Ces fonctions comprennent une fonction de calcul ou de lecture d'une capacité de multiplexage liée au premier flux de données, une fonction d'extraction d'une partie du second flux de données. Ces fonctions seront plus particulièrement développées dans la suite de la description. De façon symétrique, les procédés de décompression de données sont mis en oeuvre par un décodeur selon l'invention.

De façon générale, le procédé décrit ci-après consiste à créer des codes de longueur fixée pour les données de source importantes (flux s_H), en attribuant plusieurs mots de codes à chaque réalisation possible de cette source.

Ainsi, pour transmettre un symbole, il est possible de choisir parmi les différentes représentations possibles de celui-ci. Ce choix, qui est une variable multi-valuée, définit une capacité de stockage qui va pouvoir être utilisée pour transmettre conjointement d'autres données (cf. exemple de la figure 5 détaillée ci-après). Ce sont les données d'importance moindre, représentées par un flux noté s_L , qui vont être représentées via la représentation multiple des symboles.

La figure 1 présente le procédé de création de ces codes, plus précisément le procédé de création de codes multiplexés.

A l'étape 100, la source importante, s_H , prend ses valeurs dans un alphabet à Ω éléments, qui peut être défini par l'expression (E1) en annexe. La loi μ de probabilité d'apparition des symboles sur cet alphabet est supposée connue. On note μ_i la probabilité associée au symbole a_i de l'alphabet de la source s_H , comme représenté par l'expression (E2).

Le procédé de création des codes, représenté sur la figure 1, peut ensuite se décomposer en 2 étapes principales:

- Pour chaque symbole a_i , choix du nombre N_i de mots de code affectés à ce symbole,
- Allocation des mots de codes aux symboles.

La première étape consiste en une sélection des paramètres de codes c et (N_i) décomposée en différentes étapes 120, 130, 140 et 150.

A l'étape 120-1, un paramètre c de longueur de mot de code, en nombre de bits, est choisi. Dans une première réalisation (en référence aux figures 2A et 5), le paramètre c prend la valeur $c = 4$. Dans une autre réalisation (en référence aux figures 2B, 7 et 10), le paramètre c prend la valeur 3. Ceci définit 2^c mots de codes à l'étape 120-2, à répartir entre les symboles de l'alphabet A.

En fonction des valeurs possibles de la probabilité μ_i à l'étape 130-1, on partitionne l'ensemble des symboles de l'alphabet A en deux sous-ensembles A_m et A_M respectivement à l'étape 130-2 et à l'étape 130-3. Le premier est l'ensemble des symboles a_i dont la probabilité μ_i est inférieure ou égale à $1/2^c$, le second est son complémentaire dans A. Les cardinaux de ces ensembles sont notés respectivement Ω_m et Ω_M .

A l'étape 140, la loi de probabilité $\tilde{\mu}$ sur les symboles de A_M est alors calculée. Elle est donnée par l'expression (E3).

A l'étape 150-1, le nombre de mots de codes par symbole est alors choisi de manière à vérifier approximativement l'expression (E4), sous la contrainte de l'expression (E5) pour le sous-ensemble A_M . Dans ce but, un algorithme classique d'optimisation peut être utilisé. A l'étape 150-2, le nombre de mots de codes par symbole est déterminé pour l'ensemble des symboles de l'alphabet A.

Dans une variante du procédé de création des codes, comprenant les étapes 120, 130, 140 et 150, une étape additionnelle est ajoutée après l'étape 150, et les étapes 120-1 et 150-1 se déclinent respectivement sous la forme suivante.

Soient $f_1 = 2, f_2 = 3, f_3 = 5 \dots, f_v$, les v premiers nombres premiers. En plus du paramètre c à l'étape 120-1, un nombre premier f_v est également choisi parmi ces nombres premiers.

A l'étape 150-1, on procède de la même manière, mais en ajoutant une contrainte supplémentaire de l'étape 150-11 pour le choix du nombre de mots de code N_i associés à chaque symbole : la décomposition en facteurs premiers de tous les N_i ne doit pas contenir de facteur premier supérieur à f_v .

Après l'étape 150-1, la décomposition en facteurs premiers de chaque N_i est alors effectuée, et on calcule pour tout N_i le nombre de fois que chaque facteur premier f_j , avec $1 \leq j \leq v$, apparaît dans cette décomposition. Ce nombre est noté $\alpha_{i,j}$, où i indexe le symbole a_i considéré et j indexe le nombre premier f_j considéré. La correspondance entre a_i et les nombres $\alpha_{i,1} \dots \alpha_{i,v}$ peut être stockée dans une table dite "table alpha".

L'étape additionnelle consiste en une allocation des mots de codes aux symboles. Cette étape se décompose en différentes étapes 160 et 180 développées ci-après.

10 A l'étape 160, un étiquetage binaire (0000, 0001, ...) des symboles rangés selon, à titre d'exemple uniquement, un ordre lexicographique est réalisé. Ceci permet d'affecter aux différents symboles N_i de mots de codes, le paramètre N_i étant un entier déterminé à l'étape précédente (150-2). L'ensemble des mots de codes ainsi associés à un symbole a_i est appelé classe d'équivalence de a_i , et notée C_i à l'étape 180.

15

On associe alors aux mots de codes de chaque classe d'équivalence une valeur entre 0 et $N_i - 1$, appelée ici état. Cette valeur identifie le mot de code au sein de la classe d'équivalence.

20

Ainsi, à chaque mot de code c_i est associé un symbole a_i et une variable d'état comprise entre 0 et $N_i - 1$, comme illustré par l'expression (E6).

25

Un exemple de code ainsi construit est donné dans la table A, figure 2A pour $f_v = 5$ et $c = 4$. La table A comprend pour chaque symbole a_i de la colonne 13-A, une colonne 11-A comprenant les classes C_i , une colonne 12-A comprenant les mots de code associés à chaque classe, une colonne 14-A comprenant les nombres N_i de mots de code par classe, une colonne 15-A comprenant les probabilités μ_i associées à chaque classe, une colonne 16-A comprenant un état q de chaque mot de code d'une classe. Dans cet exemple, les symboles a_i comprennent 4 éléments de l'alphabet.

30

Une variante de code construit est donnée dans la table B, figure 2B pour $f_v = 2$ et $c = 3$. Cette table B définit un code multiplexé binaire, c'est-à-dire un code multiplexé tel que pour tout symbole a_i , le cardinal N_i de la classe d'équivalence associée est une puissance entière de 2. En notant l_i cette puissance, le cardinal N_i de la classe d'équivalence vérifie E(11). La

- table B comprend pour chaque symbole a_i de la colonne 13-B, une colonne 11-B comprenant les classes C_i , une colonne 12-B comprenant les mots de code associés à chaque classe, une colonne 14-B comprenant les nombres N_i de mots de code par classe. Dans cet exemple, la variable d'état q n'existe pas mais d'autres indications sont précisées et leur utilité sera plus
- 5 largement comprise à la lecture de la description : une colonne 15-B comprend les probabilités μ_i associées chacune à une classe, une colonne 18-B comprend un nombre D_i de bits pouvant être stockés, D_i étant associé à chaque classe, une colonne 17-B comprend un jeu \overline{U}_i de D_i bits pour chaque mot de code d'une classe. Dans cet exemple, les symboles a_i comprennent 4 éléments de l'alphabet avec i prenant les valeurs de 1 à 4.
- 10 La condition (E11) sur les cardinaux des classes d'équivalence mène au fait que le choix d'un mot de code au sein d'une classe d'équivalence permet de stocker un nombre de bit entier égal au logarithme en base 2 du cardinal de la classe d'équivalence. Ce nombre de bits D_i peut donc s'écrire selon (E12). D_i représente une capacité de multiplexage d'un symbole donné. Les tables A et B sont des tables de codage, appelées aussi tables de mots de code
- 15 multiplexés ou table de code multiplexé.
- Selon une troisième variante de réalisation, un code multiplexé binaire peut également être construit à partir d'un arbre binaire de codage associé à un préfixe de mot de code, ce préfixe étant de longueur variable et inférieure à une longueur maximale étant la hauteur de l'arbre,
- 20 tel que le code de Huffman présenté sur la figure 7. La relation "inférieure" est à prendre dans le sens "inférieur ou égal". L'arbre se divise d'abord en deux branches prenant respectivement la valeur 1 et la valeur 0. De manière récurrente, chaque branche de l'arbre est divisée en deux branches prenant respectivement la valeur 0 et la valeur 1. Le paramètre c est donné par la hauteur de l'arbre, c'est-à-dire par la longueur du préfixe le plus long
- 25 (dans l'exemple $c = 3$) de l'arbre. Une classe d'équivalence C_i est définie par un ensemble de mots de code de longueur fixe ayant chacun une première partie en commun, appelée préfixe commun. Ce préfixe commun est la partie du mot de code à longueur variable utilisée pour représenter le symbole a_i et symbolisée sur la figure 7 par le trajet formé par les branches successives en trait plein. Pour certains trajets en trait plein représentant un
- 30 symbole a_i , il reste un certain nombre de trajet en trait tireté représentant chacun une deuxième partie du mot de code appelée suffixe. Le suffixe d'un mot de code définit le jeu de bit U_i comme indiqué sur la figure par une flèche.

Ainsi, l'arbre de codage est défini comme un module de codage, mettant en correspondance les symboles du flux de données prioritaire et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole. Ces mots de code sont de longueur fixe et comprennent une première et une deuxième partie de mot de code, par exemple un préfixe et un suffixe de longueurs variables. Dans une variante, la première et la deuxième partie de mot de code peuvent correspondre respectivement au suffixe et au préfixe du mot de code. De manière plus générale, un mot de code peut comprendre plusieurs parties de code.

10 Les figures 2 et 3 illustrent le procédé de codage, appelé également d'encodage.

Le processus d'encodage se décompose de la manière suivante:

- 15 - étapes 1: le flux de données d'importance moindre S_L de l'étape 1-1 est encodé en une suite binaire $b = (b_1, b_2, \dots, b_{K_B})$ en utilisant un encodage réversible à l'étape 1-2. A cet effet, un codeur réversible de type Huffman ou codeur arithmétique (non restrictif) peut être utilisé. Cela aboutit à la génération d'une suite de bits, notée b à l'étape 1-3.
- 20 - étapes 2 : de la séquence de symboles s_1, s_2, \dots, s_{K_H} du flux s_H à l'étape 2-1 et de la lecture de table des mots de code multiplexés à l'étape 2-2, on déduit les valeurs n_1, n_2, \dots, n_{K_H} associées à l'étape 2-3. Par exemple, pour un symbole s_1 correspondant au symbole a_i d'une table de mots de codes multiplexés, n_1 prend la valeur de N_i correspondant à la réalisation du symbole a_i .
- 25 - étapes 3: on en déduit la valeur Λ , ici selon l'expression (E7) à l'étape 3-1. On calcule le nombre K'_B de bits qui vont pouvoir être stockés en utilisant la redondance intrinsèque des codes multiplexés, selon l'expression (E8) à l'étape 3-2. Ce nombre K'_B de bits représente la capacité de multiplexage du flux s_H pour ce processus d'encodage.
- 30 - étapes 4: à condition que $K_B < K'_B$, à l'étape 4-1, K'_B bits du flux b , par exemple les K'_B derniers bits du flux b (étapes 4-2, 4-3) sont utilisés pour calculer (étape 4-4) un entier long γ (étape 4-5), donné ici par la relation (E9). Ceci correspond à la transformation des K'_B

derniers bits du flux binaire en une variable globale. Si la condition $K_B < K'_B$ n'est pas vérifiée à l'étape 4-1, le procédé reprend à l'étape 2-1 en réduisant la valeur K_H (étape 4-11).

5 - étapes 5 : la valeur γ permet alors de calculer les états q_t , $1 \leq t \leq K_H$ (étape 5-2), en utilisant par exemple une méthode de décomposition euclidienne généralisée (étape 5-1), comme illustré dans l'algorithme (A1) annexé. Il s'agit d'une génération de flux d'états q_t , q_t étant une variable multivaluée.

10 - étapes 6 : pour tout t tel que $1 \leq t \leq K_H$, la connaissance du symbole s_t et de l'état q_t calculé à l'étape précédente permet de choisir le mot de code dans la table des mots de codes multiplexés (étape 6-1). On obtient le flux multiplexé m comprenant les mots de code m_1 à m_{K_H} (étape 6-2).

15 - étapes 7 : les $K_B - K'_B$ bits du flux d'importance moindre (étape 7-1) sont alors concaténés à la séquence de mots de codes multiplexés précédemment construite (étape 7-2) pour former le flux transmis (étape 7-3).

20 Au moins l'étape 3 est mise en oeuvre par la fonction de calcul du module de traitement. Les étapes 4, 5 et 6 au moins sont mises en oeuvre par la fonction d'extraction du module de traitement.

De façon générale, pour un procédé d'encodage, un processus de décodage associé est effectué en procédant aux opérations inverses de celles de l'encodage.

25 Une variante du procédé d'encodage est illustrée sur la figure 4 et permet d'éviter les calculs sur entiers longs. La variante du processus d'encodage se décompose de la manière suivante:

- étapes 21 : elles correspondent aux étapes 1 de la figure 3.

30 - étapes 22 : elles correspondent aux étapes 2 de la figure 3.

- étapes 23: le nombre total de fois où chaque facteur premier f_j apparaît dans l'ensemble des décompositions en facteurs de la séquence de variables n_i est alors déterminée à partir de la

table dite "alpha" (étapes 23-1 et 23-2). Il est noté d_j dans la suite, et représente le nombre de variables f_j -valuées qui peuvent être multiplexées avec le flux s_H . Ainsi, pour chaque facteur premier f_j , d_j représente la somme des α_{ij} de la séquence de variables n_i .

5 - étapes 24 : on choisit alors les transformations qui vont être utilisées pour transformer le train binaire en ces variables f_j -valuées (étape 24-1). Ces transformations dépendent de la valeur de f_v choisie. Les transformations utilisées pour $f_v = 5$ sont présentées dans la table C de la figure 5.

10 Elles se présentent sous la forme illustrée dans les expressions (E10) annexées.

Ainsi, chaque transformation T_z prend u_{Tz} bits en entrée (que l'on notera par la suite u_T par simplification pour un z donné) et les transforment en respectivement $v_{T,1}, v_{T,2}, \dots, v_{T,v}$ variables 2, 3, 5, . . . , f_v -valuées. Dans l'exemple de la table C pour $f_v = 5$, chaque

15 transformation T_z de la colonne 31 prend u_T bits de la colonne 32 en entrée et les transforment en respectivement $v_{T,1}, v_{T,2}, v_{T,3}$ variables 2, 3, 5 -valuées des colonnes 33, 34, 35. Or le nombre requis de variables de chaque type est connu: pour chaque type de variable f_j , il est d_j (cf. étape 23-2).

20 L'algorithme A2 annexé peut être utilisé pour calculer le nombre g_{Tz} de fois que la transformation T_z doit être utilisée (étape 24-2), et ceci pour une variable z allant de 0 à z_{\max} . (Les transformations sont supposées être classées dans l'ordre décroissant de leur pertinence dans la table).

25 - étape 25 : K_B , le nombre de bits multiplexés, est calculé en effectuant le produit du nombre de fois g_{Tz} qu'une transformation T_z doit être utilisée avec le nombre de bits u_T en entrée de la transformation et en additionnant ces produits pour toutes les transformations z utilisées. Ce nombre K'_B de bits représente la capacité de multiplexage du flux s_H pour ce processus d'encodage. Cette étape 25 correspond aux étapes 3 de la figure 3. Les étapes suivantes 26-1,

30 26-2, 26-3, et 26-11 correspondent aux étapes 4-1, 4-2, 4-3 et 4-11 de la figure 3.

- étapes 27 : ayant choisi le nombre de transformations de chaque type qui seront utilisées, on les applique à la fin du flux binaire b (étape 27-1).

Pour chaque transformation T_z , les u_T bits en entrée sont vus comme la représentation binaire d'un entier e .

Cet entier est alors décomposé en plusieurs variables f_j -valuées, comme précisé dans les expressions (E10). On note $e_{r,j}$ ces variables, où :

j indique que la valeur obtenue est la réalisation d'une variable f_j -valuée, et r indique le numéro de variable f_j -valuée.

Les valeurs des $e_{r,j}$ peuvent s'obtenir à partir de e avec le procédé de l'algorithme A3.

10 Cet algorithme est réitéré un nombre g_{Tz} de fois pour chaque transformation T_z .

A l'issue de cette étape 27-1, les résultats obtenus se présentent sous la forme des expressions (E10) et sont concaténés de manière à obtenir v séquences de variables disponibles (étape 27-2):

15 - la première, noté F_1 , est une séquence de longueur d_1 de variables 2-valuée (bits),
 - la j -ème, noté F_j , est une séquence de longueur d_j de variables f_j -valuée. Des pointeurs de positions, appelés t_j , sont associés aux séquences, ils sont initialement positionnés sur le début de chaque séquence.

20 - étapes 28 : à partir de ces variables, on calcule le flux d'état (étape 28-1) dont le résultat est (étape 28-2) :

$$q = (q_1, q_2, \dots, q_{K_H}).$$

Pour effectuer ce calcul, on peut procéder de la manière suivante :

25 - pour tout t tel que $1 \leq t \leq K_H$, et ainsi pour chaque symbole s_t , la décomposition en facteurs premiers de n_t permet de déterminer le nombre $\alpha_{t,j}$ de variables de chaque type (2-valuées, ..., f_j -valuées, ..., f_v -valuées, j variant de 1 à v). Chaque séquence F_j précédemment obtenue est divisée en K_H segments successifs comprenant $\alpha_{t,j}$ bits pour t variant de 1 à K_H . Le procédé est réitéré pour j variant de 1 à v . Chaque variable n_t -valuée (q_t) est obtenue par
 30 le procédé réciproque de la décomposition euclidienne itérative, appliquée aux segments $F_{t,j}$ de variables f_j -valuées. Un exemple de l'implémentation de ce procédé est décrit par l'algorithme A4. Notons qu'à la fin de ces étapes 28, toutes les variables des flux F_j ont été utilisées.

- étapes 29 : pour tout t tel que $1 \leq t \leq K_H$, la connaissance du symbole s_t et de l'état q_t calculé à l'étape précédente permet de choisir le mot de code dans la table des mots de codes multiplexés (étapes 29-1). Le flux multiplexé m est alors obtenu (étape 29-2).

- 5 - étapes 30 : les $K_H - K'_H$ bits du flux d'importance moindre (étapes 30-1) sont alors concaténés à la séquence de mots de codes multiplexés précédemment évaluée (étape 30-2). On obtient le flux transmis (étape 30-3).

Au moins l'étape 25 est mise en oeuvre par la fonction de calcul du module de traitement.

- 10 Au moins l'étape 27 est mise en oeuvre par la fonction d'extraction du module de traitement.

Les procédés d'encodage présentés en référence aux figures 2, 3 et 4 peuvent être généralisés selon le procédé d'encodage de la figure 8 :

- 15 - à partir de la table de mots de code multiplexés et de la séquence de symboles s_H , les valeurs n_1, n_2, \dots, n_{KH} associées sont calculées de manière à calculer la capacité de multiplexage K'_B de la séquence de symboles s_H à l'étape I.
- le flux b pré-encodé est divisé en deux parties b' et b'' à l'étape II en fonction de la capacité de multiplexage K'_B ,
- la partie b' du flux est transformée en une suite d'états q en utilisant les valeurs n_1, n_2, \dots, n_{KH} à l'étape V
- 20 - à partir de cette suite d'états q et de la table de mots de codes multiplexés, les mots de code multiplexés sont choisis à l'étape VII,
- ces mots de code sont assemblés pour former un flux multiplexé m à l'étape VIII,
- la partie concaténée b'' du flux b est concaténée avec le flux multiplexé m à l'étape IX.

25

Au moins l'étape I est mise en oeuvre par la fonction de calcul du module de traitement.

L'étape II au moins est mise en oeuvre par la fonction d'extraction du module de traitement.

- 30 Un exemple de création d'une capacité de stockage selon l'invention est illustré sur la figure 5. Ainsi, pour chaque symbole s_t du flux de données s_H est attribuée une classe correspondante C_t et les mots de code associés $c_{t,q}$ en fonction d'une table de codage. Chaque état q_t d'un flux de données q peut être conjointement stocké après le choix du mot de code c_{t,q_t} dans la table de mots de code multiplexés.

Dans le cas de la conversion du train binaire de priorité moindre, la variante du procédé d'encodage avec $f_v = 2$ peut être utilisée. Une autre variante décrite ci-après en référence à la figure 9 peut avantageusement être utilisée. La table B de la figure 2B est utilisée dans cet exemple.

5

La partie du procédé de conversion du train binaire se résume donc aux étapes suivantes :

- étape 40: elle correspond aux étapes 1 de la figure 3.
- 10 - étape 42 : de la lecture du symbole s_t et de la lecture de table de codes multiplexés binaires, on déduit la valeur D_t associée. Cette valeur D_t correspond au nombre de bits qui peuvent être conjointement codés avec s_t . D_t est une capacité de multiplexage d'un symbole donné.
- étape 44: les D_t prochains bits du train binaire b pré-encodé sont lus. Il est noté que le train
- 15 binaire est lu au fur et à mesure en fonction d'un pointeur de positionnement. Ces D_t prochains bits sont notés $\overline{u_t}$ et jouent le même rôle que les états (q_t).
- étape 46 : le mot de code $C_{st, \overline{u_t}}$ est choisi dans la table des codes multiplexés binaires en fonction du symbole s_t et des bits $\overline{u_t}$, la table étant indexée par a_t et $\overline{U_t}$. Ce mot de code est émis sur le canal,
- 20 - étape 48 : pour chaque symbole s_t du flux s_H , t variant de 1 à K_H les étapes 42 à 46 sont effectuées.

20

- Au moins l'étape 42 est mise en oeuvre par la fonction de calcul du module de traitement.
- 25 L'étape 44 au moins est mise en oeuvre par la fonction d'extraction du module de traitement.

- A titre d'exemple d'application du procédé de la figure 9, on considère la séquence de plus
- haute priorité à transmettre $s_H = a_H a_2 a_2 a_3 a_2 a_1 a_2 a_4 a_1 a_2$, de longueur $K_H = 10$, et le train
- 30 binaire pré-encodé de faible priorité $b = 01010101010$. Le nombre de bits qui peuvent être multiplexés avec chaque réalisation de s_H est donné, pour t variant de 1 à K_H , par $(d_t) = (1, 2,$

2, 0, 2, 1, 2, 0, 1, 2). Le nombre de bits d_t du train binaire \mathbf{b} est lu au fur et à mesure pour t variant de 1 à K_H de manière à obtenir les séquences $\overline{u_t}$ de bits $(\overline{u_1}, \dots, \overline{u_{K_H}}) =$

(0, 10, 10, \emptyset , 10, 1, 01, \emptyset , 0, 10). Alors, pour tout t , le couple $(a_t, \overline{u_t})$ indexe un mot de code dans la table des codes multiplexés binaires. Le train binaire effectivement transmis

5 est 000 100 100 110 100 001 011 111 000 100.

En variante, le procédé de la figure 9 peut également utiliser l'arbre de codage. Dans cette variante, les étapes 42 à 46 se présente sous la forme suivante :

10 - étape 42 : de la lecture du symbole s_t et de la lecture de l'arbre binaire de codage, on obtient le préfixe du mot de code pour le symbole s_t . Du nombre de bits de ce préfixe, on déduit le nombre de bits D_t qui peuvent être conjointement codés avec s_t pour former une suite de bits d'une longueur totale égale à la hauteur de l'arbre de codage. D_t est une capacité de multiplexage d'un symbole donné.

15 - étape 44: les D_t prochains bits du train binaire \mathbf{b} pré-encodé sont lus. Il est noté que le train binaire est lu au fur et à mesure en fonction d'un pointeur de positionnement. Ces D_t prochains bits sont notés $\overline{u_t}$.

20 - étape 46 : le mot de code émis sur le canal résulte de la concaténation du préfixe du mot de code pour le symbole s_t et des bits $\overline{u_t}$ du train binaire \mathbf{b} . Ainsi, l'utilisation du train binaire \mathbf{b} permet de faire un choix parmi les mots de code possibles indiqués en trait tiretés sur l'arbre de codage de la figure 7 pour un symbole donné.

25 L'étape 44 au moins est mise en oeuvre par la fonction d'extraction du module de traitement.

A titre d'exemple, en utilisant la séquence s_H et le train binaire \mathbf{b} indiqués précédemment dans le cas de la figure 10 et l'arbre de codage de la figure 7 pour déterminer les préfixes des

mots de code, il est obtenu le flux m_i de mots de code par concaténation des préfixes et des suffixes $\overline{u_i}$.

De manière générale, l'arbre de codage permet de définir un préfixe de mots de code pour chaque symbole du flux s_H , ce qui revient à définir plusieurs mots de code possibles pour certains symboles. Le choix parmi ces mots de code possibles sera fait une fois faite la lecture du code binaire pour la détermination du suffixe du mot de code et la formation du mot de code par concaténation du préfixe et du suffixe. Le calcul de la somme de l'ensemble des D_i associés aux symboles formant le flux s_H permet de déterminer la capacité de multiplexage du flux s_H .

D'autres variantes du procédé d'encodage sont illustrées ci-après en référence aux figures 11 et 12.

Il peut être intéressant de considérer de créer un code multiplexé non pas sur l'alphabet, mais sur un "alphabet produit". On appelle "alphabet produit", un alphabet constitué non pas de symboles mais de séquences de symboles. Dans l'exemple de la figure 11, la source s_H comprend K symboles. Elle est convertie en une source de C-uplet notée H (de longueur K/C), ces C-uplets se dénommant $H_1, H_2, H_3, \dots, H_{K/C}$ et respectivement numérotés 51, 52, 53 et 55. Tout C-uplet de symboles a la probabilité d'apparition (E13). Il en est déduit le calcul de la distribution de probabilité μ_H associé aux C-uplets. La création de l'arbre binaire de codage (appelé "arbre binaire de codage produit") est effectuée en considérant, pour chaque C-uplet, les probabilités d'apparition de chaque séquence de longueur C données en (E13). Le préfixe du mot de code associé à une séquence de symboles est lu sur l'arbre de codage.

25

D'après la figure 11, pour chaque séquence de symboles, la fonction de multiplexage regroupe un certain nombre de fonctions effectuant les étapes correspondant au procédé en variante de la figure 9. Dans chaque étape, le "symbole" est remplacé par une "séquence de symboles". Ainsi, le procédé d'encodage utilisant un arbre de codage est appliqué

directement sur les réalisations de C-uplets de la source H .

30

Si la taille de l'alphabet A est trop importante pour pouvoir utiliser un arbre de codage, il est également possible de remplacer "l'arbre de codage produit" par un code arithmétique comme illustré sur la figure 12. Ainsi, la source s_H 70 est découpée en C -uplets, ce qui conduit à un nombre de C -uplets égal à K/C . Ces C -uplets peuvent être relativement longs et sont encodés par des codeurs arithmétiques (non restrictif) indépendants. Dans l'exemple de la figure 12, chaque C -uplet est encodé par un codeur arithmétique distinct 71, 72, 73 et 75. La sortie des ces codeurs arithmétiques sont des suites $H_1, H_2, H_{K/C}$ de bits de longueur variables numérotés 81-1, 82-1, 85-1. La longueur c des mots de codes correspond à la plus longue suite de bits H_t possible en sortie des codeurs arithmétiques. Chaque suite de bits est alors vue comme un préfixe d'un mot de code. Pour chaque préfixe de longueur inférieure strictement à la longueur c , il existe plusieurs mots de code de longueur c correspondant au même symbole.

La formation d'un mot de code est la même que dans la variante de "l'alphabet produit". Ainsi, le flux binaire encodé b numéroté 90 est lu au fur et à mesure pour former les suffixes 81-2, 82-2, 85-2 de façon à compléter les préfixes 81-1, 82-1, 85-1 et former les mots de code multiplexés. Si le nombre K/C n'est pas un entier, les $C' = K - C \lfloor K/C \rfloor$ derniers symboles forment un C' -uplet qui est encodé arithmétiquement.

Comme indiqué pour la figure 9, du nombre de bits d'un préfixe, on déduit le nombre de bits D_t ($1 < t < K$) qui peuvent être conjointement codés avec H_t pour former une suite de bits d'une longueur totale égale à la hauteur de l'arbre de codage. D_t est une capacité de multiplexage d'une séquence de symboles donnée. Le calcul de la somme de l'ensemble des D_t associés aux séquences de symboles formant la source H permet de déterminer la capacité de multiplexage du flux s_H .

De manière générale, un codeur arithmétique ou un arbre de codage permet d'établir un préfixe de mots de code pour chaque séquence de symboles, ce qui revient à définir plusieurs mots de code possibles pour certaines séquences de symboles. Le choix parmi ces mots de code possibles sera fait une fois faite la lecture du code binaire pour la détermination du suffixe du mot de code et la formation du mot de code par concaténation du préfixe et du suffixe.

Ainsi, l'invention permet le multiplexage de deux flux de données S_H et S_L , afin de diminuer la sensibilité aux erreurs de l'un d'eux S_H , désigné comme plus important ou prioritaire. Ces deux flux peuvent être distingués dans la même source de signaux, notamment comme dans les quelques exemples suivants de sources S_H et S_L :

5

- basses fréquences et hautes fréquences extraites par décomposition multi-résolution (par bancs de filtres, transformées en ondelettes) d'un signal,

10

- information de texture (ex : coefficients DCT, coefficients ondelettes) et information de mouvement,

- bits de poids forts et bits de poids faibles des coefficients ondelettes ou des échantillons quantifiés d'un signal.

15

Bien entendu, l'énumération ci-dessus n'a aucun caractère exhaustif.

20

Par ailleurs, dans la mesure où les mots de code sont de longueur fixe (ou bien si l'on utilisait des marqueurs de synchronisation), l'invention permet la création d'un code multiplexé permettant de décrire conjointement deux flux, dont un au moins bénéficie d'une synchronisation parfaite.

Annexe 1 - Formules

$$(E1) \quad \mathcal{A} = \{a_1, \dots, a_i, \dots, a_\Omega\}$$

$$(E2) \quad \mu_i = P(a_i)$$

$$(E3) \quad \tilde{\mu}_i = \frac{2^c}{2^c - \Omega_M} \mu_i$$

$$(E4) \quad N_i = (2^c - \Omega_m) * \tilde{\mu}_i$$

$$(E5) \quad \sum_{i \in \mathcal{A}} N_i = 2^c$$

$$(E6) \quad c_{i,j} \Leftrightarrow (s_i, q_j)$$

$$(E7) \quad \Lambda = \prod_{t=1}^{K_H} n_t$$

$$(E8) \quad K'_B = \lfloor \log_2(\Lambda) \rfloor$$

$$(E9) \quad \gamma = \sum_{r=1}^{K_H} b_{r+K_B-K'_B} 2^{r-1}.$$

$$(E10) \quad u_{\mathcal{T}} \text{bits} \Leftrightarrow \begin{cases} v_{\mathcal{T},1} \text{ variables 2-valuée,} & e_{1,1}, e_{2,1}, \dots, e_{v_{\mathcal{T},1},1} \\ v_{\mathcal{T},2} \text{ variables 3-valuée,} & e_{1,2}, e_{2,2}, \dots, e_{v_{\mathcal{T},2},2} \\ \dots & \\ v_{\mathcal{T},\nu} \text{ variables } f_{\nu_i}\text{-valuée,} & e_{1,\nu}, e_{2,\nu}, \dots, e_{v_{\mathcal{T},\nu},\nu} \end{cases}$$

$$(E11) \quad \forall i \in [1..\Omega], \exists l_i \in \mathbb{N} / N_i = 2^{l_i}$$

$$(E12) \quad D_i = \log_2(N_i) = c - l_i$$

$$(E13) \quad P(S_t S_{t+1} \dots S_{t+C-1}) = P(S_t) P(S_{t+1}/S_t) \dots P(S_{t+C-1}/S_{t+C-2})$$

Annexe 2 - Algorithmes

A1

```

 $\gamma' = \gamma$ 
Pour  $t = 1 : K_H$ 
   $q_t = \gamma' \text{ modulo } n_t$ 
   $\gamma' = \frac{\gamma' - q_t}{n_t}$ 
Fin pour

```

A2

```

 $z = 0$ 
% Tant qu'il reste des variables  $f_j$ -valuées à obtenir
Tant que  $\text{sum}(d_j) > 0$ 
  % Calcul du nombre de fois que la transformation  $\mathcal{T}_z$  est utilisée
   $g_{\mathcal{T}_z} = \text{floor}(\min(\frac{d_j}{v_{\mathcal{T}_z,j}}))$  où  $v_{\mathcal{T}_z,j} \neq 0$ 
  % Calcul du nombre de variables  $f_j$ -valuée
  % qui n'ont pas été transformées par la transformation  $\mathcal{T}_z$ 
  Pour chaque  $j$  entre 1 et  $\nu$ 
     $d_j = d_j - g_{\mathcal{T}_z} * v_{\mathcal{T}_z,j}$ 
    % Essaie la transformation suivante
   $z = z + 1$ 

```

A3

```

 $e' = e$ 
Pour  $j = 1 : \nu$ 
  Pour  $r = 1 : v_{\mathcal{T},j}$ 
     $e_{r,j} = e' \text{ modulo } f_j$ 
     $e' = \frac{e' - e_{r,j}}{f_j}$ 
  Fin pour
Fin pour

```

```

    Pour  $j = 1 : \nu$ 
       $tj = 1$ 
    Fin pour

    Pour  $t = 1 : K_H$ 
       $q_t = 0$ 
A4  Pour  $j = \nu : 1$  par -1
      Pour  $r = 1 : \alpha_{t,j}$ 
         $q_t = q_t * f_j + F_j(t_j)$ 
         $tj = tj + 1$ 
      Fin pour
    Fin pour
  Fin pour

```

Revendications

1. Codeur de compression de données numériques, caractérisé en ce qu'il comprend:
 - une entrée pour un premier flux de données (S_H), et un second flux de données (S_L),
 - 5 - un module de codage, établissant une correspondance entre des symboles du premier flux de données et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole, et
 - un module de traitement pour coder les symboles du premier flux de données d'après la correspondance, en choisissant parmi les mots redondants, en fonction d'une partie au moins
 - 10 du second flux de données.
2. Codeur selon la revendication 1, caractérisé en ce que les mots de code sont de longueur fixe.
- 15 3. Codeur selon l'une des revendications 1 et 2, caractérisé en ce que le module de traitement comprend:
 - une fonction de calcul de la capacité de multiplexage courante du premier flux de données (S_H), au vu du module de codage, et
 - une fonction d'extraction, dans le second flux de données (S_L), d'une partie multiplexée,
 - 20 déterminée d'après la capacité de multiplexage courante, pour être portée par lesdits mots redondants.
4. Codeur selon l'une des revendications précédentes, caractérisé en ce qu'il comprend une transformation d'un flux binaire en un flux de variable multi-valuée.
- 25 5. Codeur selon la revendication 4, caractérisé en ce qu'il comprend une transformation d'un flux binaire en un flux de variable multi-valuée, en particulier en utilisant les transformations décrites dans la table C.
- 30 6. Codeur selon la revendication 5, caractérisé en ce qu'il comprend une transformation d'un flux binaire en un flux de variable multi-valuée, en particulier en utilisant une décomposition euclidienne généralisée à partir d'une variable globale donnée par la relation (E9).

7. Codeur selon l'une des revendications 1 et 2, caractérisé en ce que le module de codage comprend une table de codage et en ce que le module de traitement comprend:

- une fonction de lecture d'une capacité de multiplexage de chaque symbole courant du premier flux de données (S_H) au vu de la table de codage, et

5 - une fonction d'extraction d'une partie du second flux de données (S_L) déterminée d'après la capacité de multiplexage, pour être portée par lesdits mots redondants.

8. Codeur selon la revendication 7, caractérisé en ce que la table de codage comprend pour chaque symbole, un nombre de mots de code associé égal à une puissance de 2.

10

9. Codeur selon l'une des revendications 1 et 2, caractérisé en ce que le module de codage comprend un arbre binaire de codage contenant, pour chaque symbole du premier flux de données, une première partie de mot de code, de longueur variable et inférieure à une longueur maximale, et en ce que le module de traitement comprend:

15 - une fonction de calcul de la capacité de multiplexage pour chaque symbole courant du premier flux de données (S_H) au vu de la première partie de mot de code de chaque symbole,
- une fonction d'extraction d'une partie du second flux de données (S_L) déterminée d'après la capacité de multiplexage, pour être portée par lesdits mots redondants.

20 10. Codeur selon la revendication 9, caractérisé en ce que chaque symbole comprend une séquence de symboles.

11. Codeur selon l'une des revendications 1 et 2, caractérisé en ce que chaque symbole comprend une séquence de symboles, en ce que le module de codage comprend un codeur arithmétique propre à calculer, pour une séquence de symboles du premier flux de données, une première partie de mot de code, de longueur variable et inférieure à une longueur maximale, et en ce que le module de traitement comprend:

25

- une fonction de calcul de la capacité de multiplexage pour chaque symbole courant du premier flux de données (S_H) au vu de la première partie de mot de code de chaque symbole,

30

- une fonction d'extraction d'une partie du second flux de données (S_L) déterminée d'après la capacité de multiplexage pour chaque symbole, pour être portée par lesdits mots redondants.

12. Codeur selon l'une des revendications 9 et 11, caractérisé en ce que ladite partie du second flux de données est concaténée à la première partie de mot de code jusqu'à la longueur maximale du mot de code.

5 13. Codeur selon l'une des revendications précédentes, caractérisé en ce que le second flux de données est préalablement encodé.

14. Codeur selon l'une des revendications précédentes, caractérisé en ce que le reste du second flux de données est concaténé aux données transmises.

10

15. Décodeur, propre à effectuer les opérations inverses de celle du codeur de l'une des revendications précédentes.

16. Procédé de compression de données numériques, caractérisé par les étapes suivantes:

- 15 a. établir une correspondance entre des symboles du premier flux de données et des mots de code, où, pour certains symboles, il existe plusieurs mots, dits redondants, correspondant au même symbole, et
- b. coder les symboles d'un premier flux de données d'après la correspondance de l'étape a., en choisissant parmi les mots redondants, en fonction d'une partie au
- 20 moins d'un second flux de données.

17. Procédé selon la revendication 16, caractérisé par des sous-fonctions conformes à l'une des revendications 1 à 14.

25 18. Procédé de décompression de données numériques, caractérisé par les étapes réciproques de celles du procédé selon l'une des revendications 16 et 17.

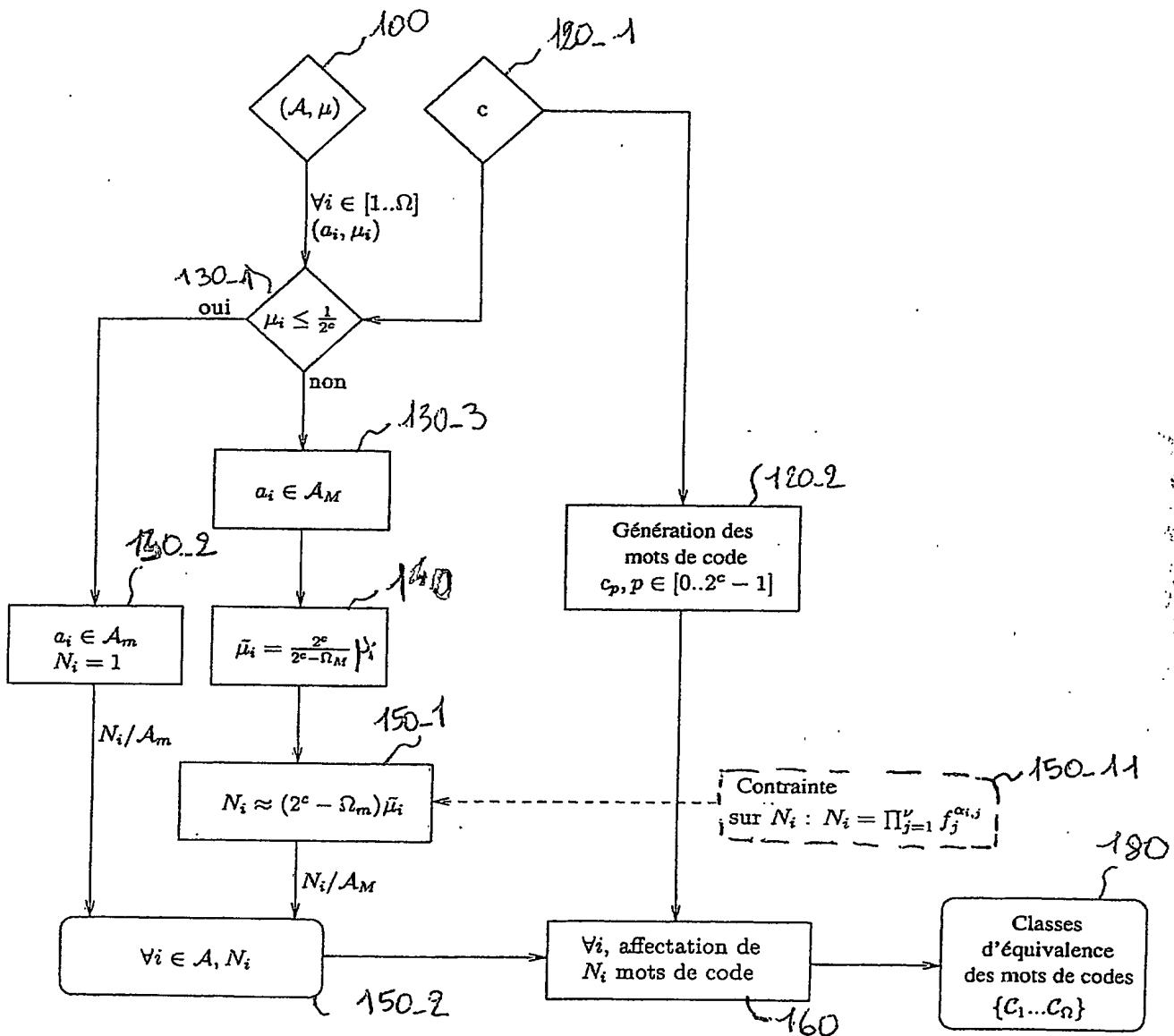


FIG. 1

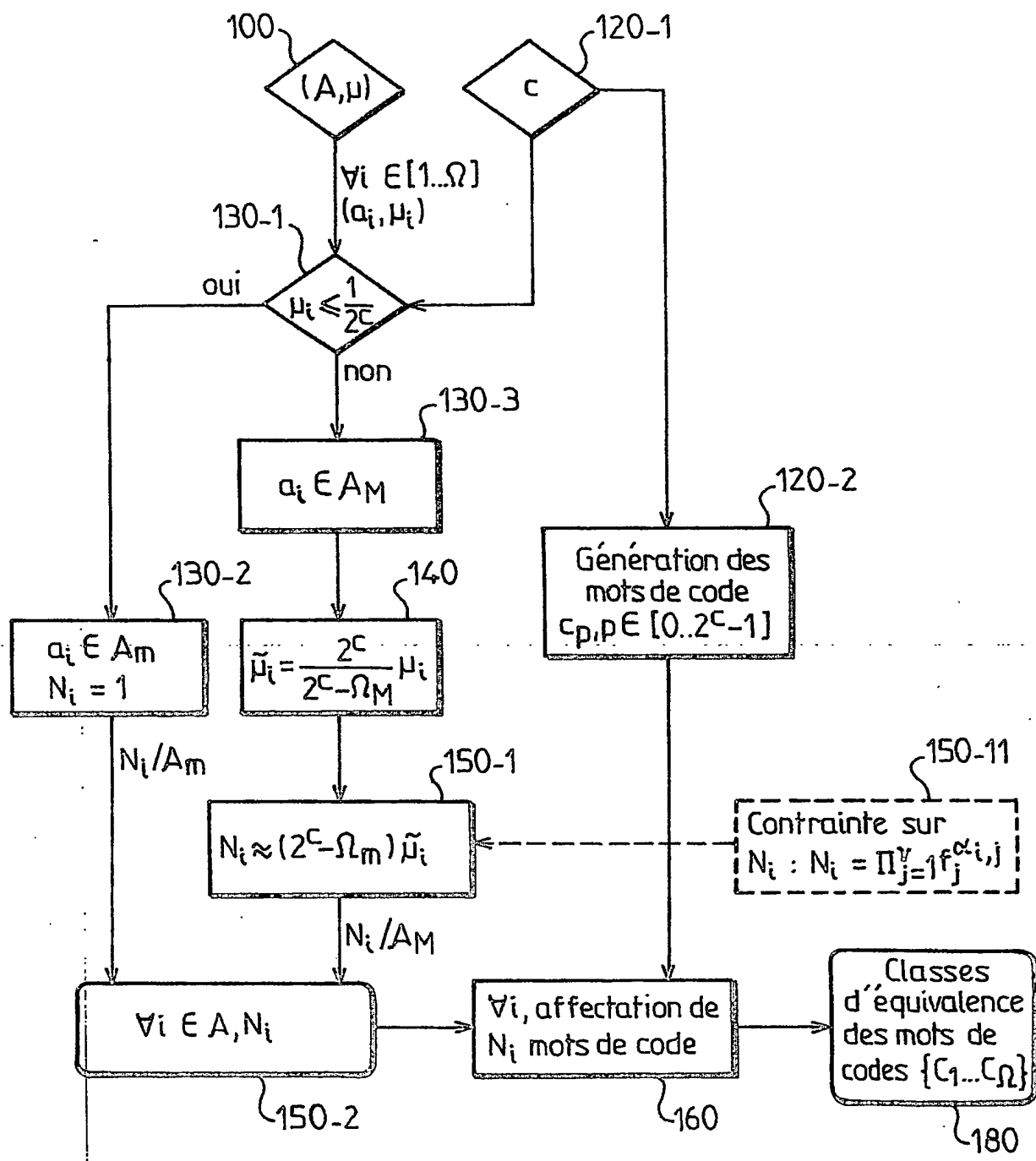


FIG.1

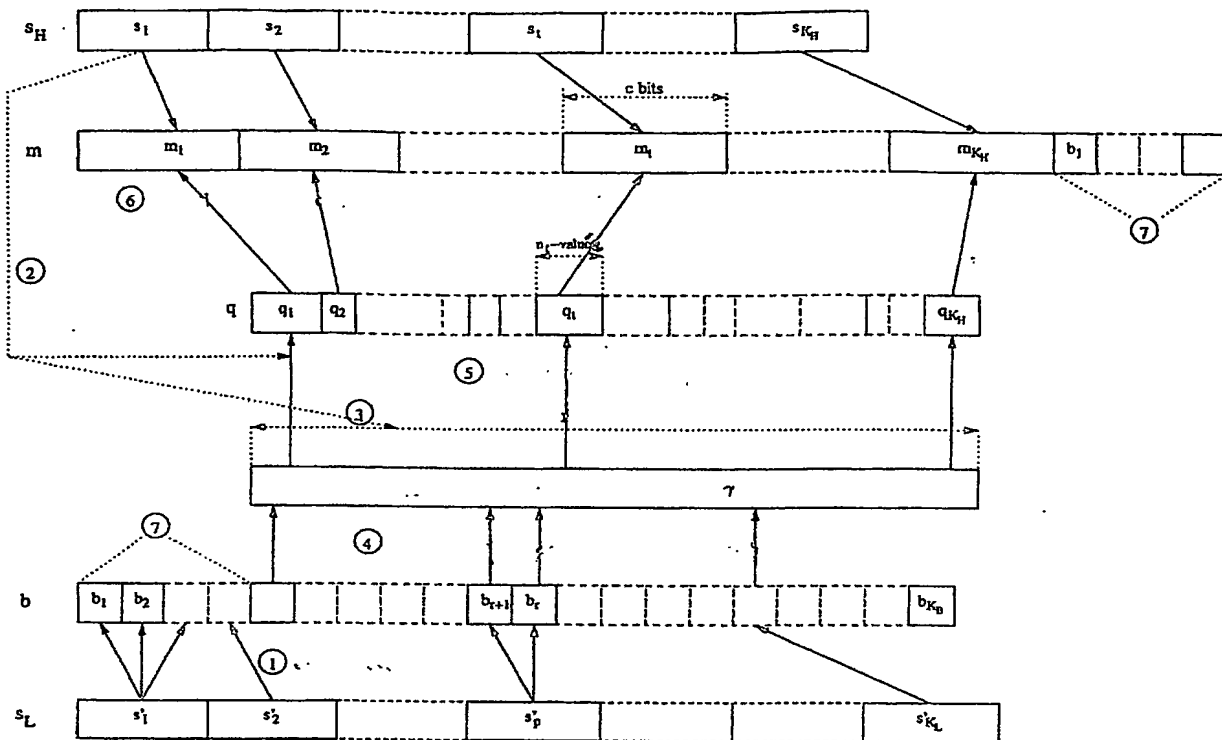


FIG. 2

11-A 12-A 13-A 14-A 15-A 16-A

classe C_i	Mot de code $c_{i,q}$	symbole a_i	$N_i = \text{card}(C_i)$	probabilité μ_i	état q
C_1	0000	a	6	0.43	0
	0001				1
	0010				2
	0011				3
	0100				4
	0101				5
C_2	0110	b	5	0.30	0
	0111				1
	1000				2
	1001				3
	1010				4
C_3	1011	c	4	0.25	0
	1100				1
	1101				2
	1110				3
C_4	1111	d	1	0.02	0

FIG 2A

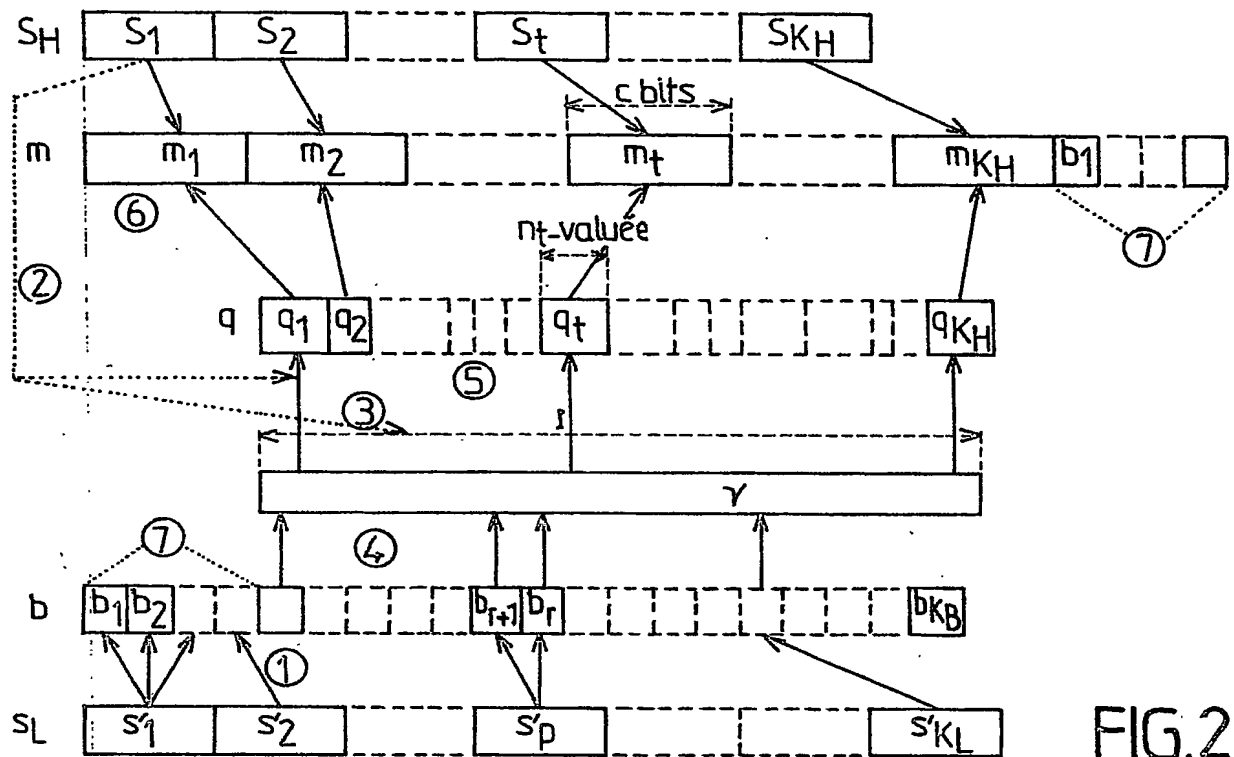


FIG.2

11-A	12-A	13-A	14-A	15-A	16-A
classe C_i	Mot de code $c_{i,q}$	symbole a_i	$N_i = \text{card}(C_i)$	probabilité μ_i	état q
A C_1	0000	a	6	0.43	0
	0001				1
	0010				2
	0011				3
	0100				4
	0101				5
C_2	0110	b	5	0.30	0
	0111				1
	1000				2
	1001				3
	1010				4
C_3	1011	c	4	0.25	0
	1100				1
	1101				2
	1110				3
C_4	1111	d	1	0.02	0

FIG.2A

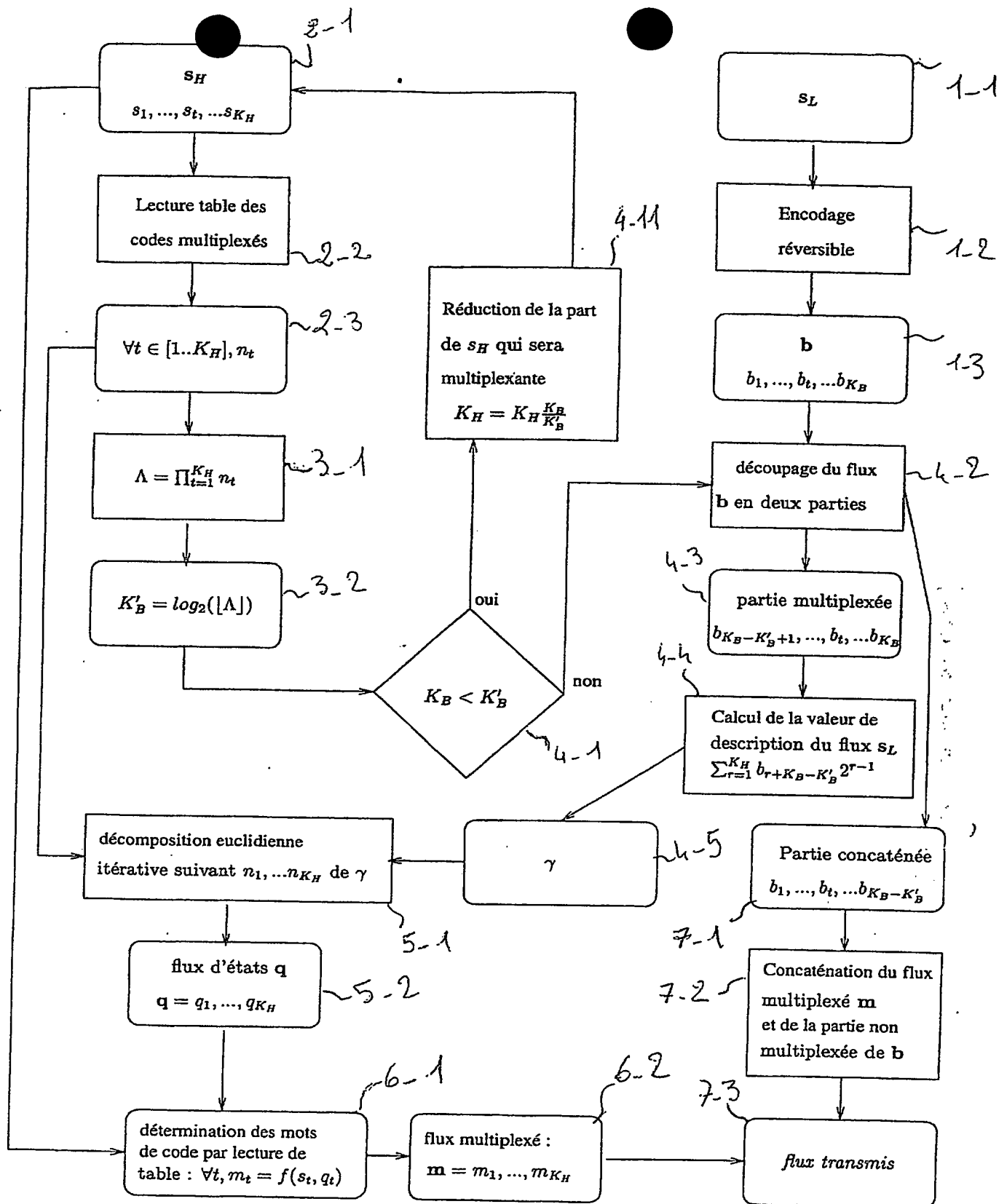


FIG. 3 -

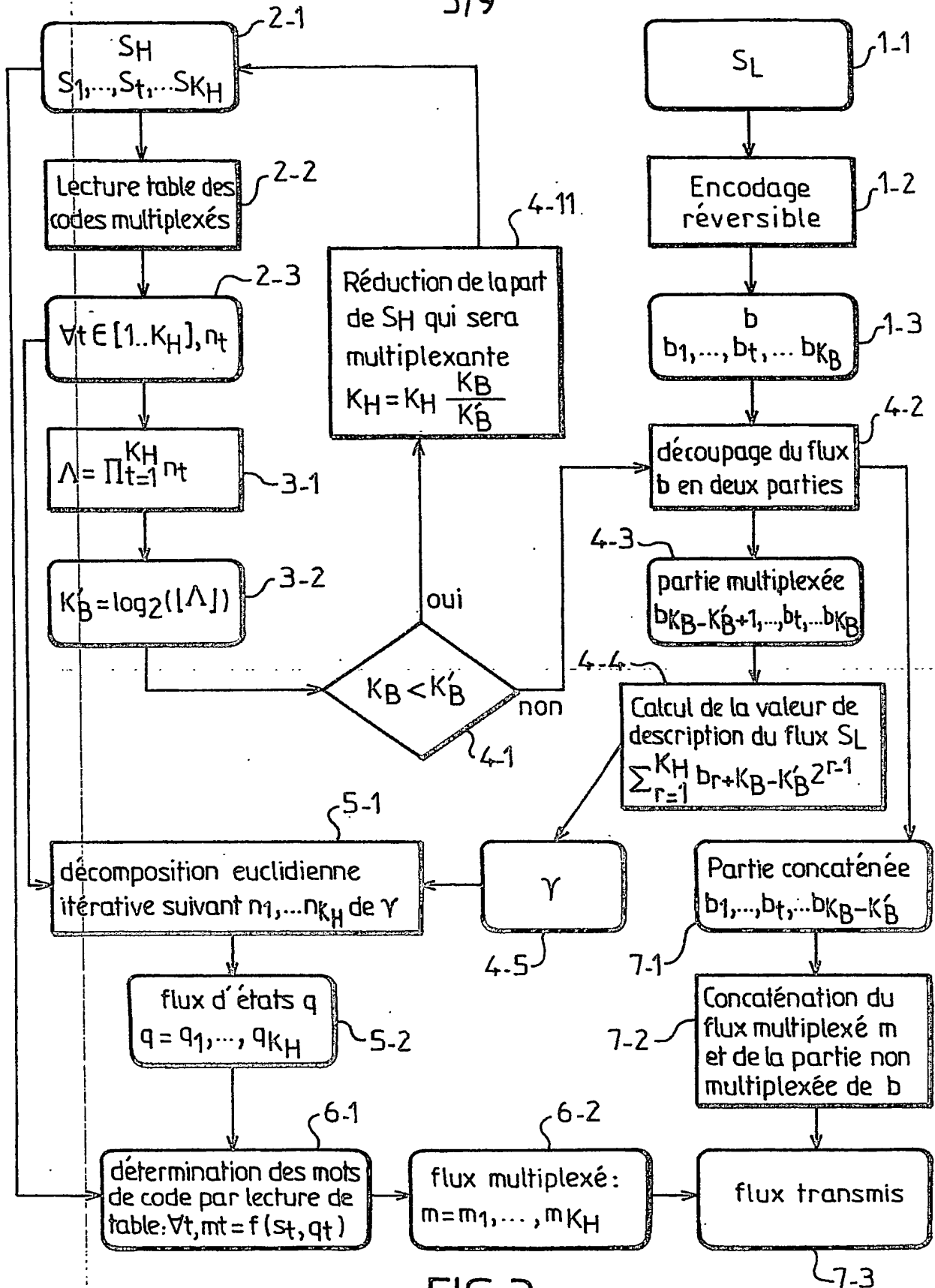


FIG.3

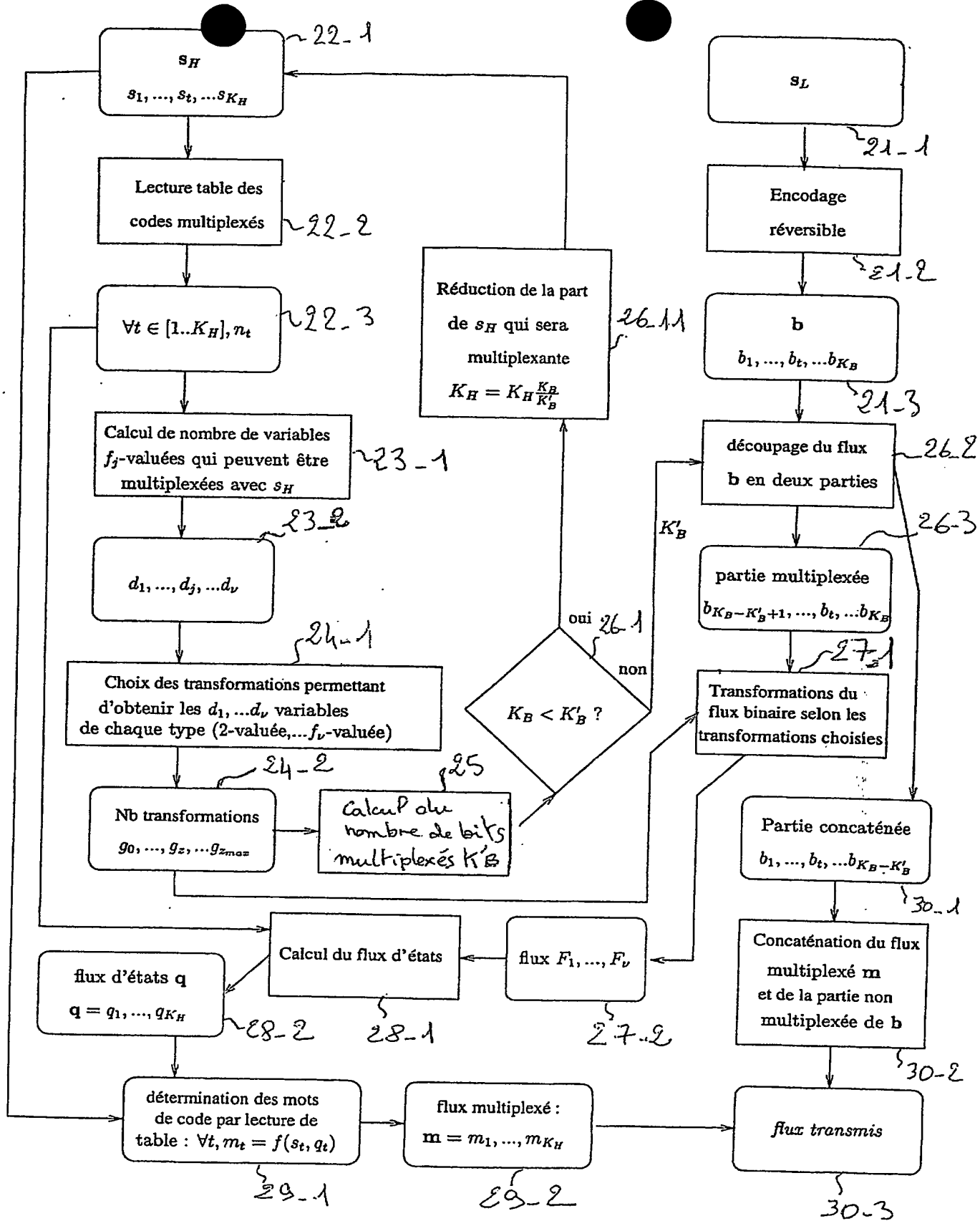


FIG. 4

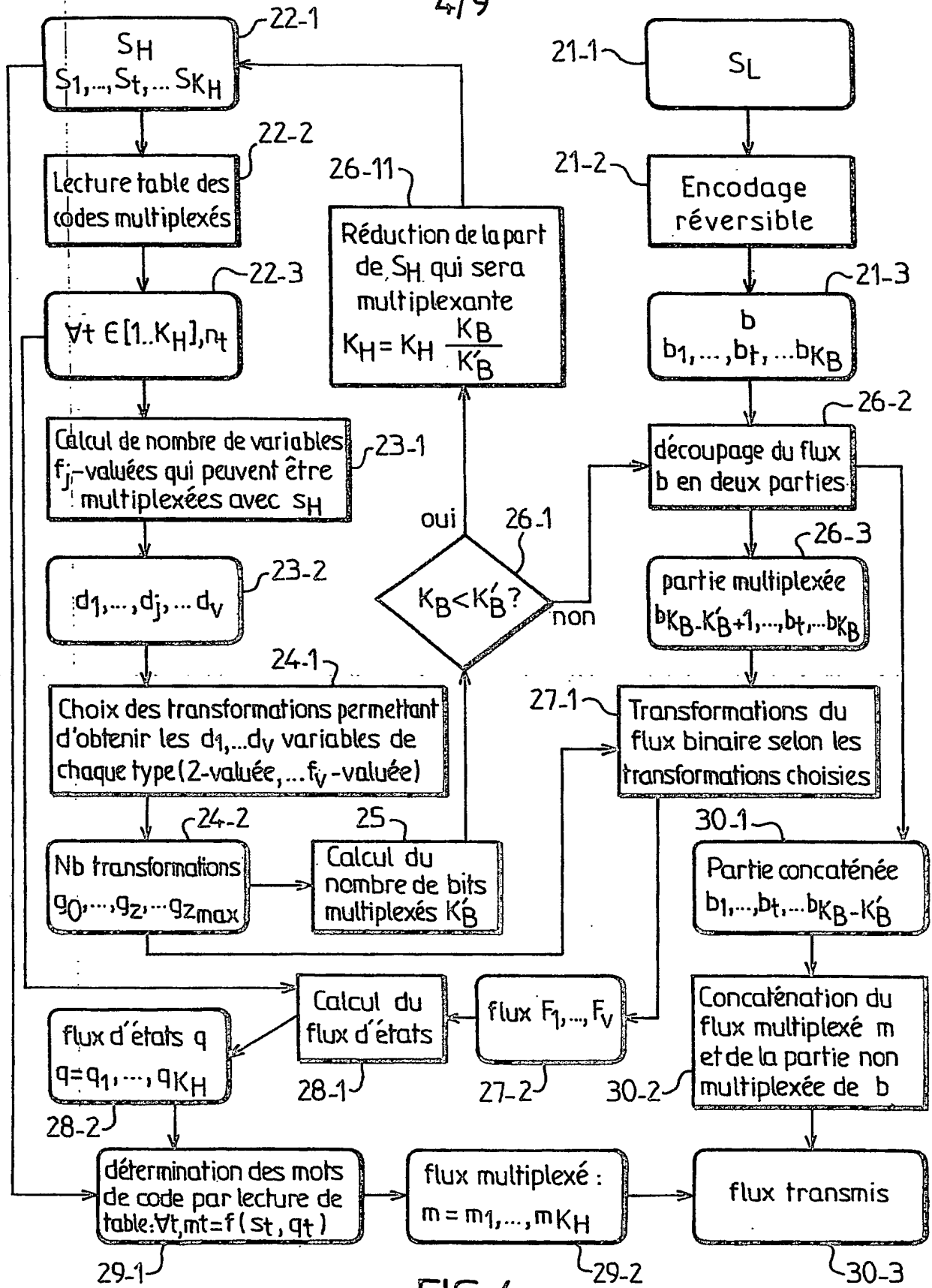


FIG.4

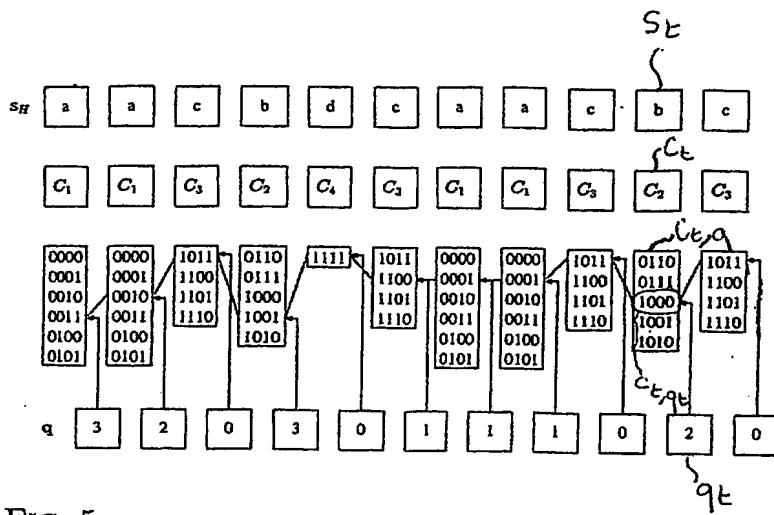


FIG. 5

31 32 33 34 35

T_z identificateur	u_{T_z}	$v_{T_z,1}$	$v_{T_z,2}$	$v_{T_z,3}$	o_{T_z}
T_0	1	1	0	0	0.0000
T_1	15	0	8	1	0.0001
T_2	21	0	3	7	0.0004
T_3	19	0	12	0	0.0010
T_4	25	0	7	6	0.0011
T_5	24	0	2	9	0.0028
T_6	14	0	3	4	0.0030
T_7	18	0	7	3	0.0034
T_8	27	0	1	11	0.0047
T_9	17	0	2	6	0.0060
T_{10}	30	0	0	13	0.0062
T_{11}	20	0	1	8	0.0080
T_{12}	11	0	7	0	0.0086
T_{13}	23	0	0	10	0.0095
T_{14}	6	0	1	2	0.0381
T_{15}	3	0	2	0	0.0566
T_{16}	2	0	0	1	0.1610
$T_{z_{max}} = T_{17}$	1	0	1	0	0.5850

FIG 6

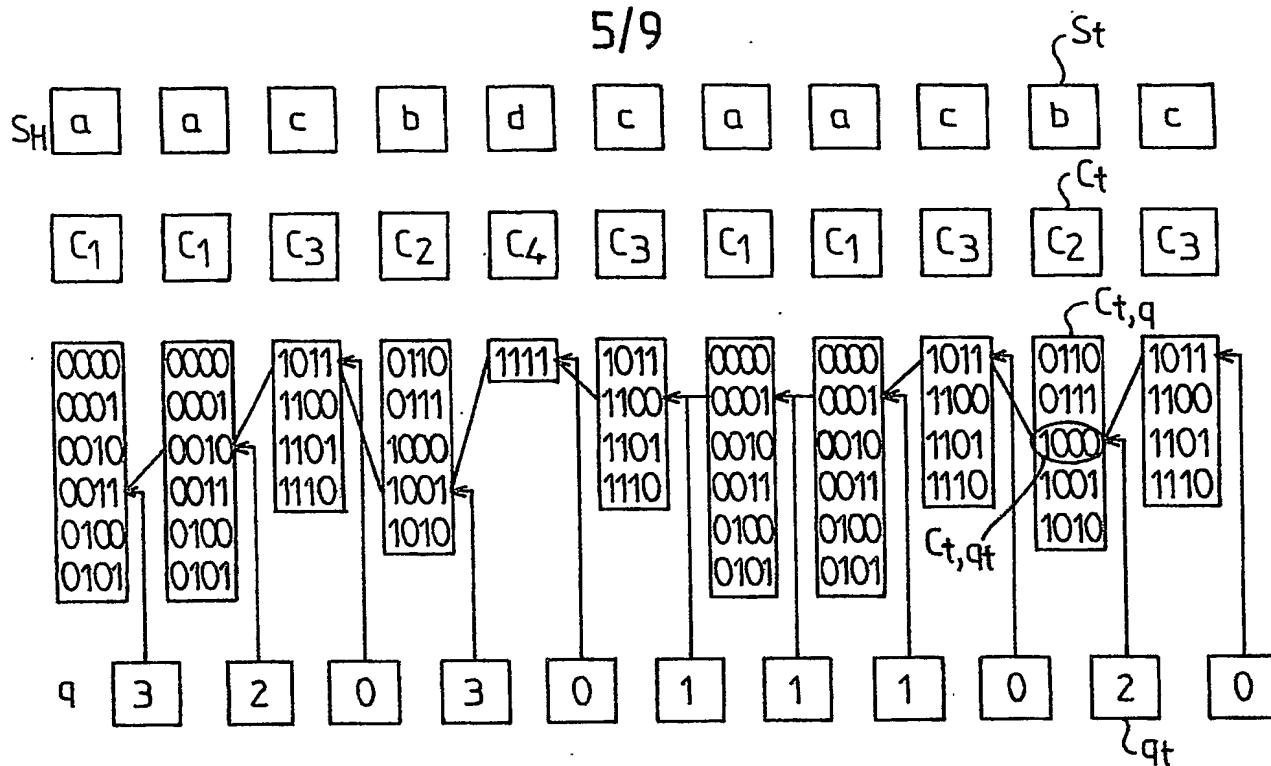


FIG. 5

	31	32	33	34	35
T_z identificateur	U_{T_z}	$V_{T_z,1}$	$V_{T_z,2}$	$V_{T_z,3}$	O_{T_z}
T_0	1	1	0	0	0.0000
T_1	15	0	8	1	0.0001
T_2	21	0	3	7	0.0004
T_3	19	0	12	0	0.0010
T_4	25	0	7	6	0.0011
T_5	24	0	2	9	0.0028
T_6	14	0	3	4	0.0030
T_7	18	0	7	3	0.0034
T_8	27	0	1	11	0.0047
T_9	17	0	2	6	0.0060
T_{10}	30	0	0	13	0.0062
T_{11}	20	0	1	8	0.0080
T_{12}	11	0	7	0	0.0086
T_{13}	23	0	0	10	0.0095
T_{14}	6	0	1	2	0.0381
T_{15}	3	0	2	0	0.0566
T_{16}	2	0	0	1	0.1610
$T_{zmax} = T_{17}$	1	0	1	0	0.5850

FIG.6

6/9

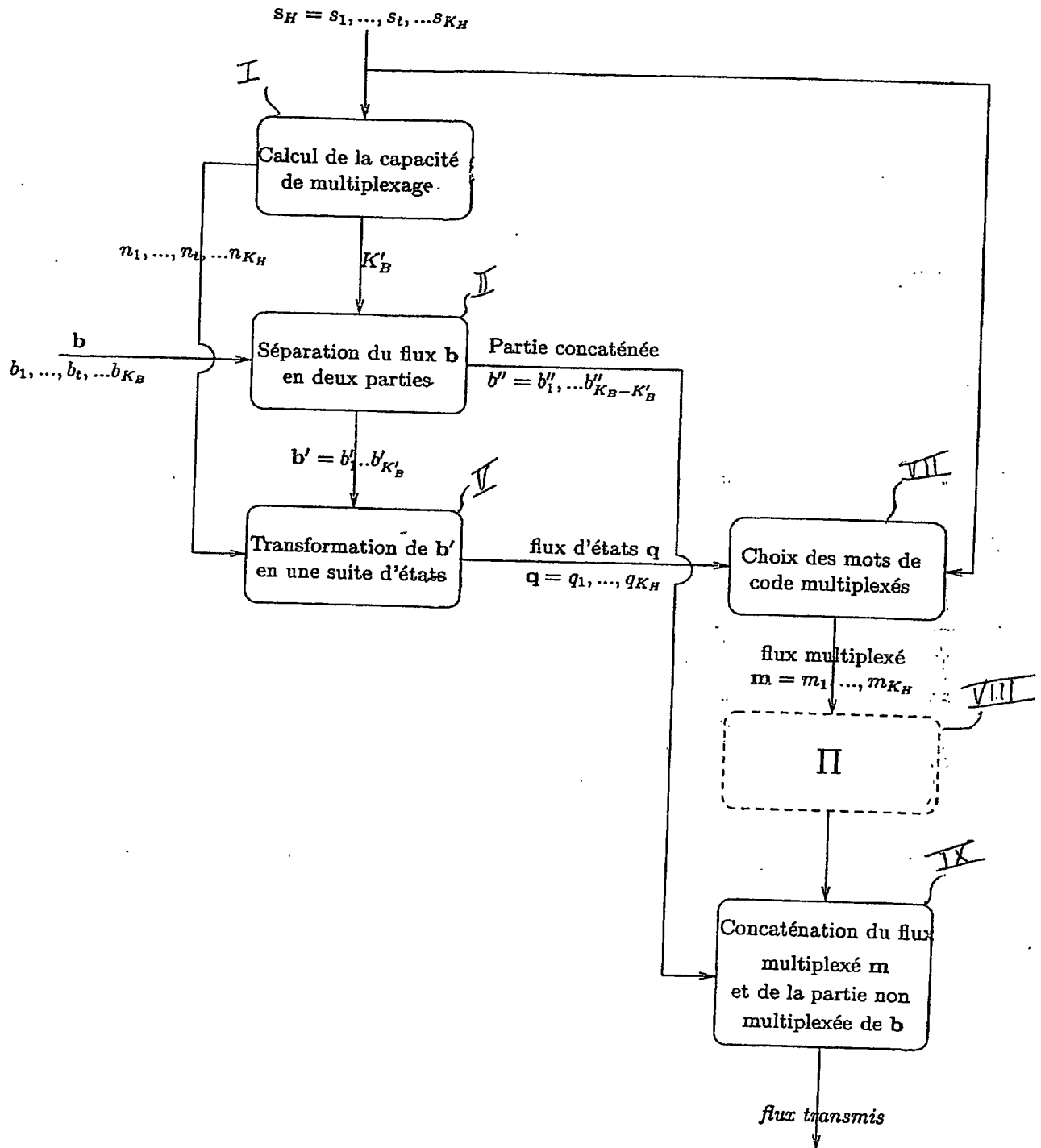


FIG 8

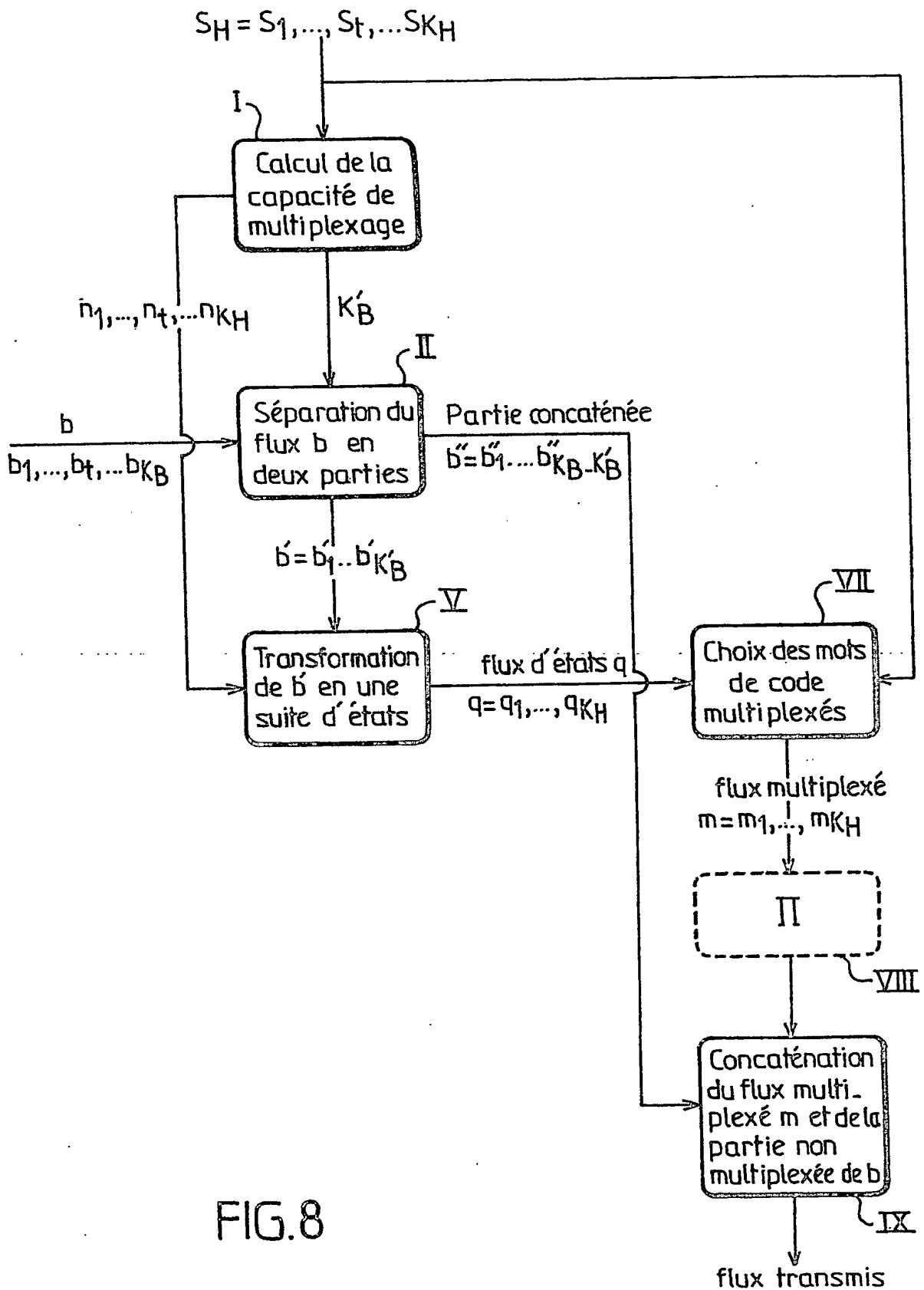


FIG. 8

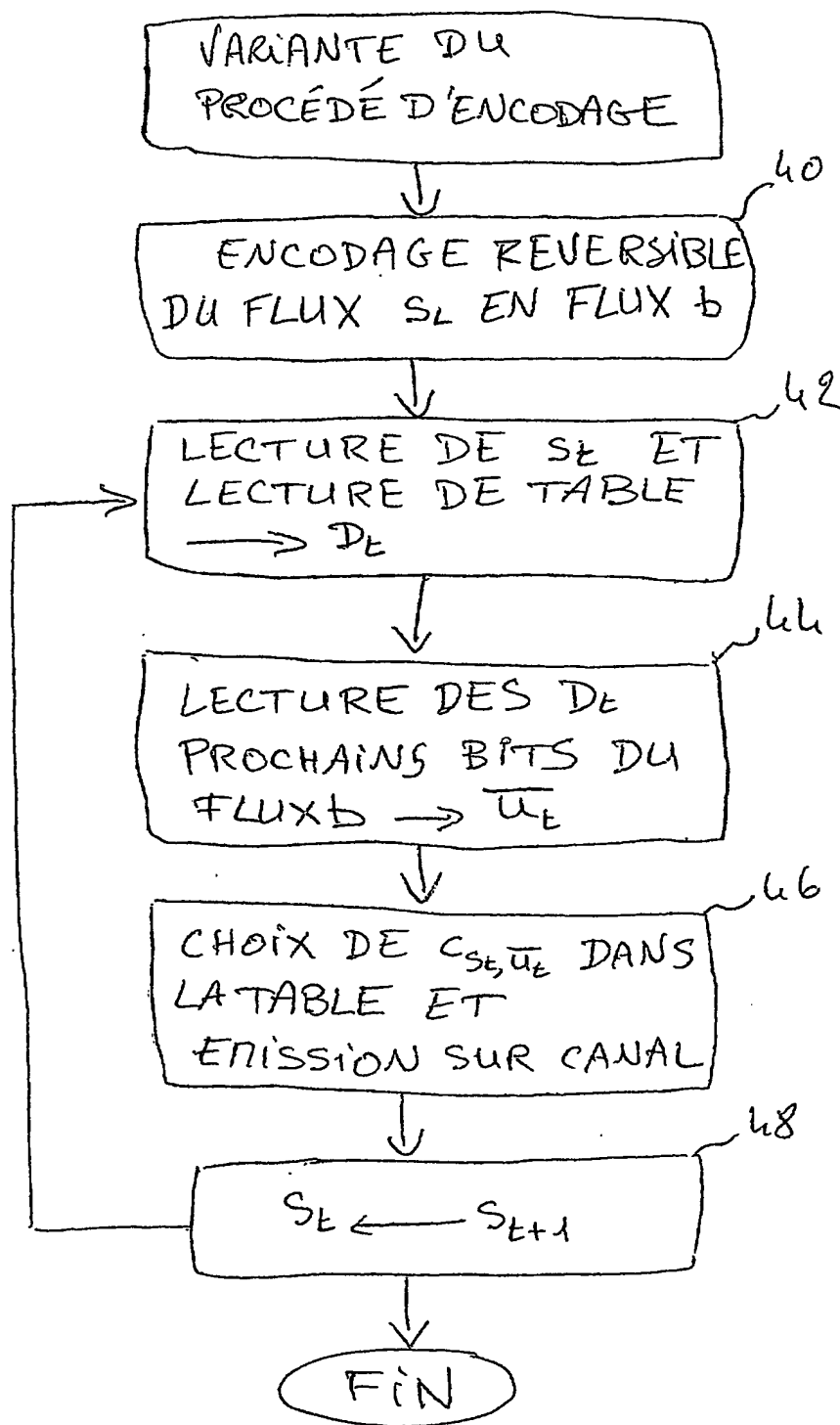


FIG 9

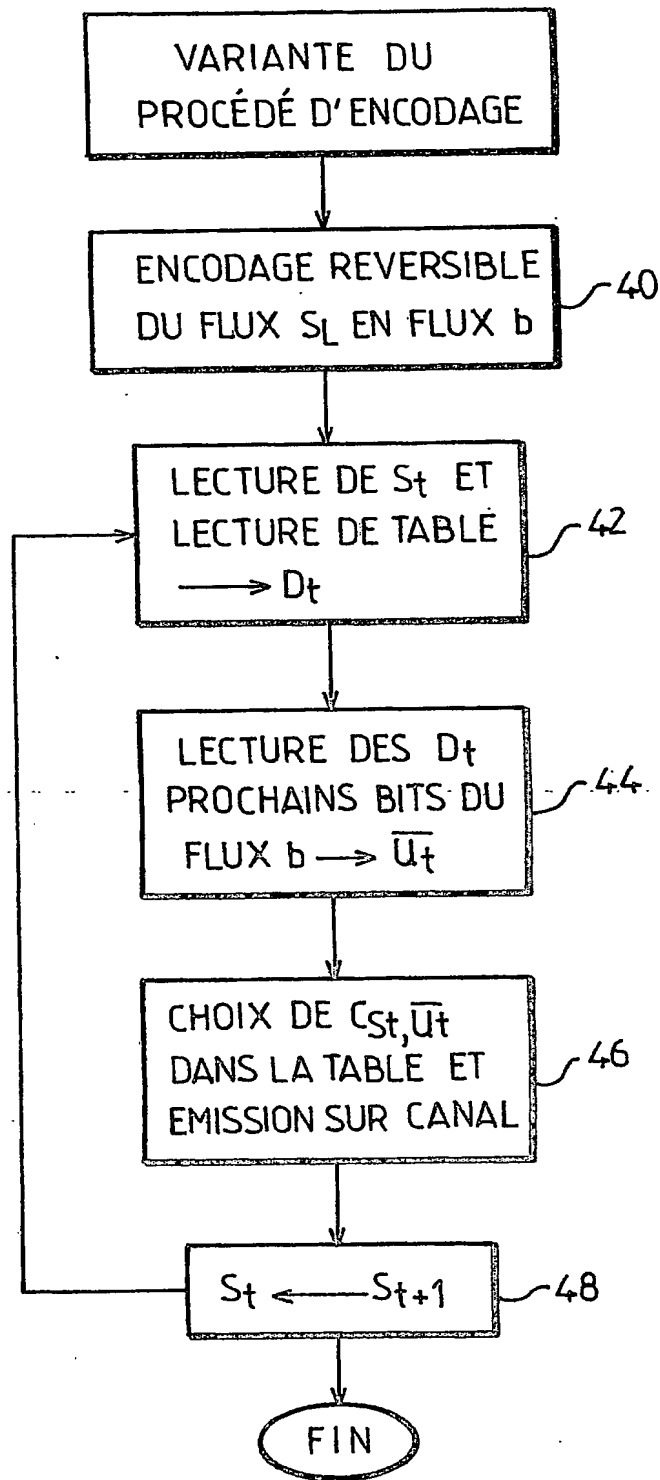


FIG.9

8/9

MAY 1966 10 00/03/00

11-B 12-B 13-B 14-B 15-B 17-B

18-B

B

class C_i	c_i, \bar{u}_i	a_i	D_i	$N_i = 2^{d_i}$	μ_i	U_i
C_1	000 001	a_1	1	2	0.30	0 1
C_2	010 011 100 101	a_2	2	4	0.43	00 01 10 11
C_3	110	a_3	0	1	0.25	\emptyset
C_4	111	a_4	0	1	0.02	\emptyset

FIG 2B

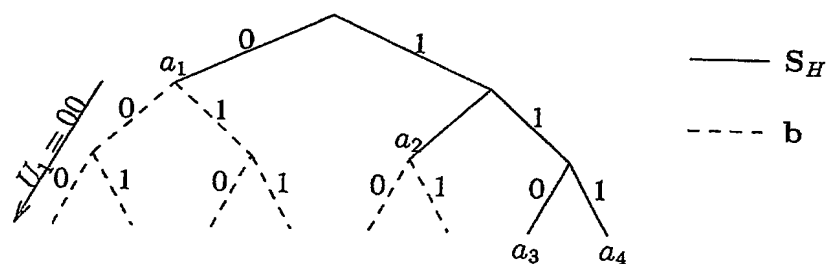


FIG 7

s_t	a_2	a_1	a_1	a_3	a_1	a_2	a_1	a_4	a_2	a_1
préfixe	10.	0..	0..	110	0..	10.	0..	111	10.	0..
suffixe \bar{u}_t	..0	.10	.1010	.1	.010	.10
m_t	100	010	010	110	010	101	001	111	100	010

FIG 10

CABINET NETTER

11-B	12-B	13-B	18-B	14-B	15-B	17-B
class C_i	c_i, \bar{U}_i	a_i	D_i	$N_i = 2^{l_i}$	μ_i	\bar{U}_i
C_1	000 001	a_1	1	2	0.30	0 1
C_2	010 011 100 101	a_2	2	4	0.43	00 01 10 11
C_3	110	a_3	0	1	0.25	\emptyset
C_4	111	a_4	0	1	0.02	\emptyset

FIG. 2B

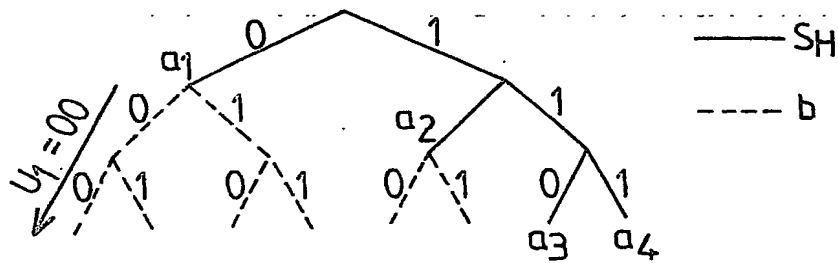
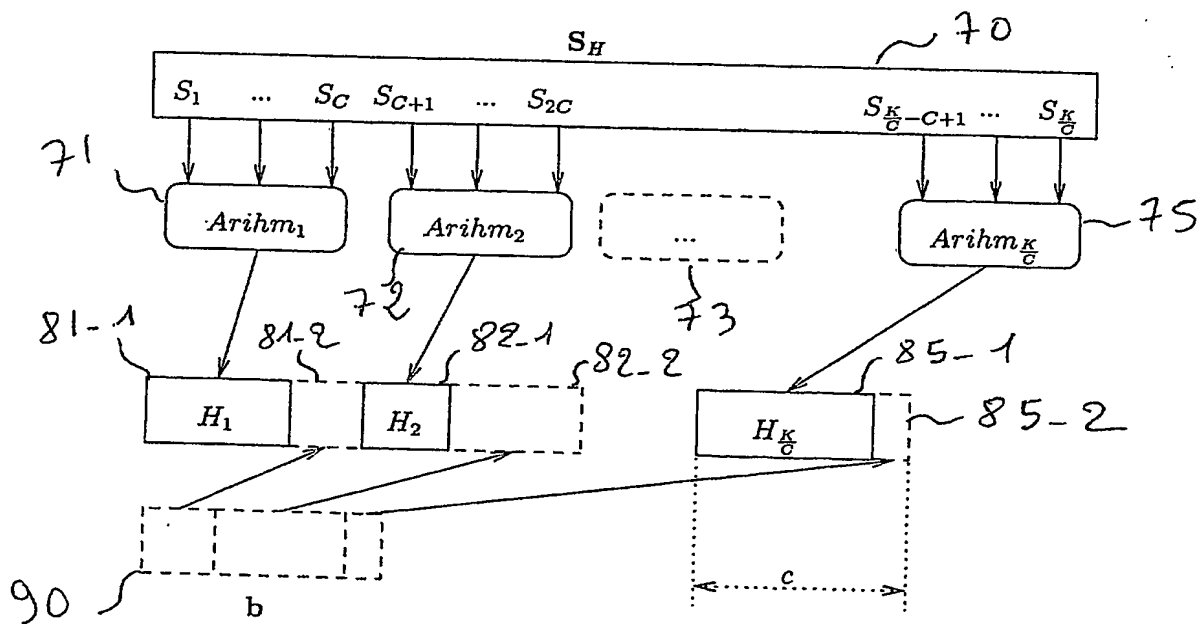
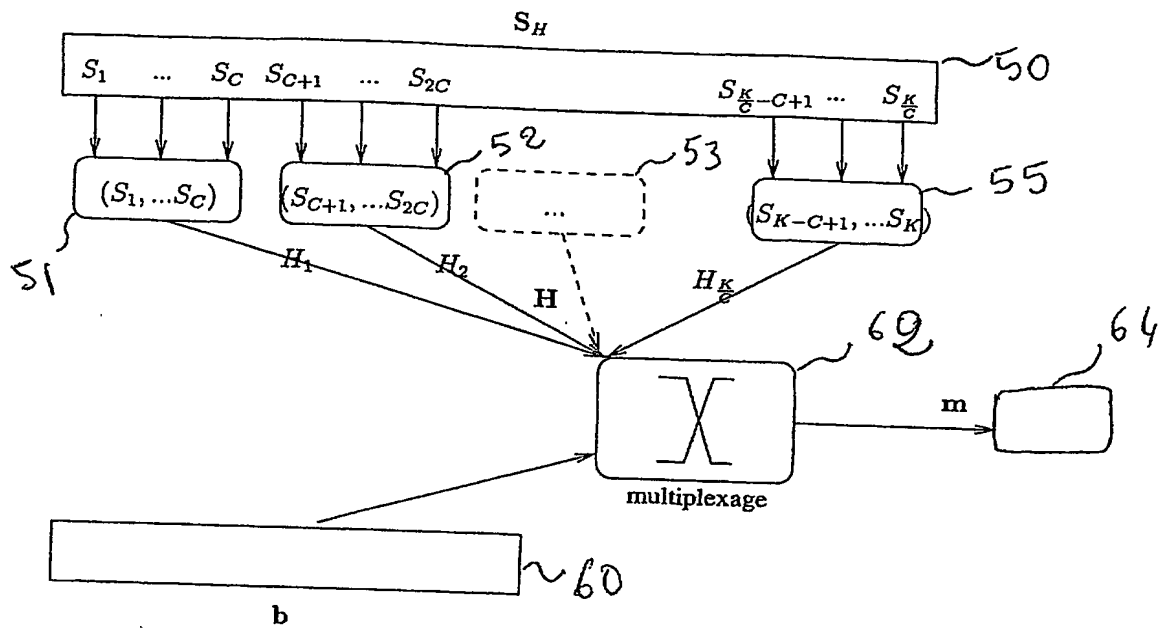


FIG. 7

St	a_2	a_1	a_1	a_3	a_1	a_2	a_1	a_4	a_2	a_1
préfixe	10.	0..	0..	110	0..	10.	0..	111	10.	0..
suffixe \bar{U}_i	..0	.10	.1010	.1	.010	.10
m_i	100	010	010	110	010	101	001	111	100	010

FIG. 10



CABINET NETTER

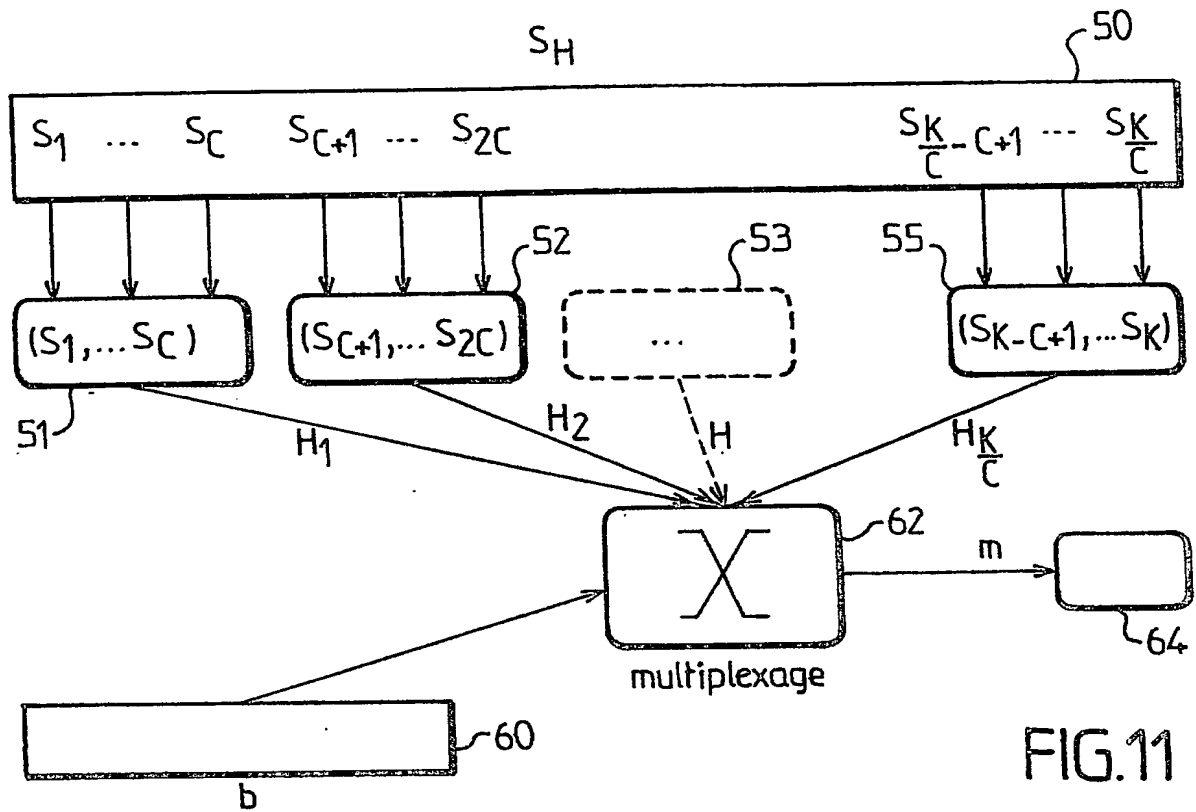


FIG.11

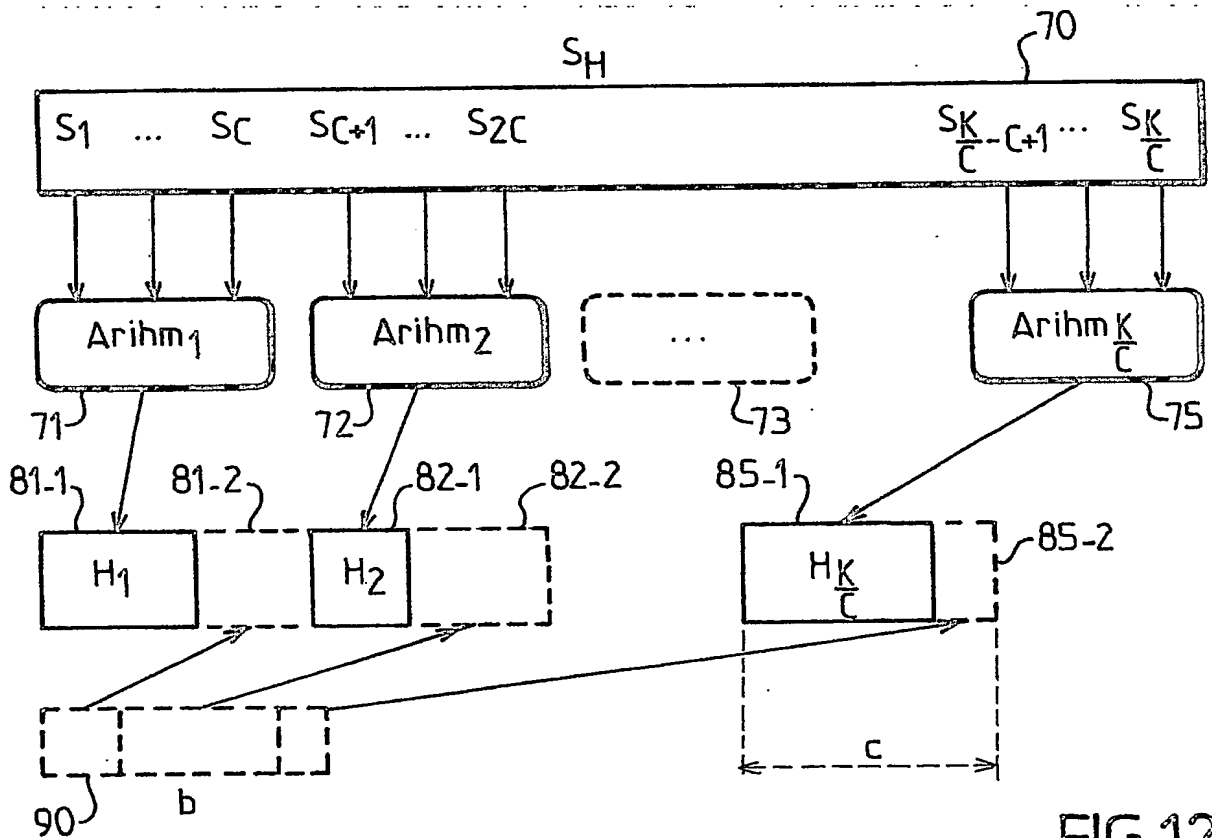


FIG.12

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08

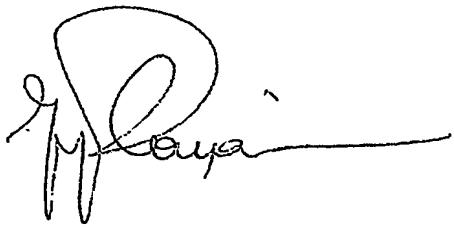
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

DÉSIGNATION D'INVENTEUR(S) Page N° 1. / 1..

(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 W / 222559

Vos références pour ce dossier (facultatif)		INRIA 59-PI (120813)	
N° D'ENREGISTREMENT NATIONAL		02/16 964	
TITRE DE L'INVENTION (200 caractères ou espaces maximum) Compression de données numériques robuste au bruit de transmission.			
LE(S) DEMANDEUR(S) : 1 - INRIA INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE 2 - Ecole Normale Supérieure de Cachan			
DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inventeurs, utilisez un formulaire identique et numérotez chaque page en indiquant le nombre total de pages).			
Nom		JEGOU	
Prénoms		Hervé	
Adresse	Rue	Le Veuleury	
	Code postal et ville	29820	BOHARS
Société d'appartenance (facultatif)			
Nom		GUILLEMOT	
Prénoms		Christine	
Adresse	Rue	2 allée Françoise Dolto	
	Code postal et ville	35135	CHANTEPIE
Société d'appartenance (facultatif)			
Nom			
Prénoms			
Adresse	Rue		
	Code postal et ville		
Société d'appartenance (facultatif)			
DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire) Paris, le 28 novembre 2002 N° Conseil 92-1197 (B) (M) Jean-Yves PLAÇAIS			

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.